WHEN CODING SKILLS BIND TO THEIR HARNESS: UNDERSTANDING AND MITIGATING CROSS-HARNESS FAILURES IN AGENTIC SFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic supervised fine-tuning (SFT) can improve an agent on the harness that generated its trajectories while degrading performance after deployment through another harness. We identify this failure in Qwen3.5-9B: SFT on Claude Code trajectories raises native success but lowers success on the held-out DeepSeek Harness by 9.8 percentage points. We call this failure *harness entanglement*. Controlled interventions reveal sensitivity to tool presentation and identifiers. We introduce **ARCHER** (**A**cross-task **R**andomized **C**onsistency for **H**arness **E**ntanglement **R**eduction), a cross-task shuffled-pair objective that trains different harness renderings with position-level loss consistency. On SWE-BENCH Verified, ARCHER reaches 62.2% and 63.2% on the held-out DeepSeek Harness and Pi, respectively, compared with 50.6% and 49.9% for the base model, while matching the base model on Claude Code and improving OpenHands. Ablations favor the shuffled construction and show higher transfer scores with position-level consistency. These results show that native-harness scores can hide deployment failures and that randomized consistency can reduce this dependence during SFT.

1 INTRODUCTION

Large language models are now used as coding agents: they inspect a repository, edit files, run tests, and react to tool output (Yao et al., 2023; Yang et al., 2024; Wang et al., 2024). A common training recipe is to distill trajectories from a stronger agent and fine-tune a student on them (Zeng et al., 2024; Chen et al., 2023; Pan et al., 2024; Agentica Project, 2025). That recipe treats the harness as a transparent carrier of the task. If the student learns the carrier as well as the task, success may disappear when a different harness is used at deployment.

Our experiments expose this failure in a controlled setting. Fine-tuning Qwen3.5-9B on CLAUDE CODE trajectories raises success on CLAUDE CODE to 51.4%, but reduces it on OPENHANDS, held-out DEEPSEEK HARNESS, and held-out PI below the base model. The loss includes tasks the base model solves under the same deployment harness, even though the fine-tuned model improves on its training harness. Native-harness evaluation therefore gives an incomplete picture of what trajectory SFT has learned. We refer to this pattern as **harness entanglement** (Figure 1, top). Recent work has asked whether trajectory SFT teaches interface surface forms in addition to task behavior (Gu et al., 2026); we measure the resulting deployment gap and test a training-time response.

Two interventions point to the interface as one part of the problem. Keeping the DEEPSEEK HARNESS runtime fixed while changing its outer skin recovers much of the native behavior. Under the first rename map, SFT-CC loses more than the content-preserving paired construction. The skin probe changes the interface as a whole; renaming targets its identifiers. The probes do not estimate additive contributions.

The training comparisons suggest a more specific remedy. A plain CLAUDE CODE +OPENHANDS mixture remains below the base model on DEEPSEEK HARNESS. Per-trajectory identifier randomization raises DEEPSEEK HARNESS above base, but gives up substantial OPENHANDS performance. The content-preserving paired regularizer reaches 50.9% on DEEPSEEK HARNESS, compared with 62.2% for the cross-task shuffled construction. These constructions differ in pairing,

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.

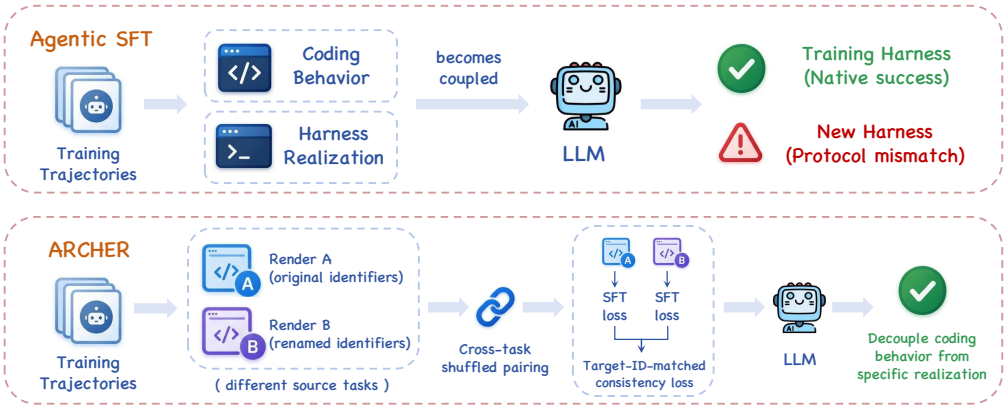


Figure 1: Harness entanglement and ARCHER. **Top:** agentic SFT couples coding behavior with a particular harness realization: native success can coexist with protocol mismatch under a new harness. **Bottom:** ARCHER cross-associates views from different source tasks, applies the ordinary SFT loss to both renderings, and adds loss consistency at target-ID-matched positions. The resulting model is evaluated on a harness excluded from its transfer-arm training data.

packing, and retained context. Within the shuffled comparisons, both consistency variants exceed the no-consistency control; the target-ID rule has a smaller mean advantage over random positions.

Training on every harness is not a general solution. Interfaces change, and a training mixture can only cover the versions observed during collection. We instead ask whether the training objective can make the student less dependent on the particular rendering that produced its data.

We propose **ARCHER** (**A**cross-task **R**andomized **C**onsistency for **H**arness **E**ntanglement **R**eduction), a cross-task shuffled-pair objective with position-level consistency. On held-out DEEPSEEK HARNESS and PI, ARCHER outperforms the random-position and shuffled-no-consistency controls while retaining the base model's performance on CLAUDE CODE and improving on OPENHANDS. The same ordering appears on both held-out interfaces.

Our contributions are:

- We document **harness entanglement**: native-harness SFT gains can coexist with a loss on other harnesses for the same coding tasks.
- We use fixed-runtime skins, identifier renaming, task-level retention, and interaction traces to locate where the loss appears. These probes also show cases in which SFT improves transfer, so the effect is not an automatic consequence of fine-tuning.
- We introduce **ARCHER**, a cross-task shuffled-pair consistency objective. Both shuffled consistency variants improve on the no-consistency control, with a smaller mean advantage for the target-ID rule over random positions.

2 RELATED WORK

Harness-based agentic systems. Agentic coding systems pair a language model with an interactive harness that mediates tool invocation and feedback (Yao et al., 2023; Yang et al., 2024; Wang et al., 2024). SWE-bench and SWE-bench Verified score task success through such harnesses (Jimenez et al., 2024; OpenAI, 2024). Recent work instead treats the harness itself as an object of study, including efficient or natural-language harness designs (Ruan et al., 2026; Ning et al., 2026; Pan et al., 2026), harness repair from failure traces (Shao et al., 2026), and harness-centric evaluation (Li et al., 2026). These works motivate treating the harness as more than fixed measurement apparatus. We study the complementary question of what happens when the deployment harness differs from the training harness. Distilling teacher trajectories into a student via SFT is the standard recipe for agentic coding models (Zeng et al., 2024; Agentica Project, 2025; Kang et al., 2025). This literature primarily evaluates the student through a fixed harness, whereas our focus is transfer

across interfaces. Prior work notes that imitation can teach surface style without transferable capability (Gudibande et al., 2024), and that SFT and RL differ in how they treat out-of-distribution behavior (Chu et al., 2025). Closest to our question, Gu et al. (2026) diagnose whether trajectory SFT teaches task semantics or interface surface forms; our study is complementary: we measure the deployment consequence of interface binding as cross-harness transfer failure, localize it with fixed-runtime interventions, and propose a training-time mitigation. We ask not whether SFT generalizes to new tasks, but whether it generalizes to new *interfaces* for the same tasks.

Interface sensitivity and robustness. Catastrophic forgetting under sequential fine-tuning is long-standing (Kirkpatrick et al., 2017), including for LLM fine-tuning (Luo et al., 2025). Our setting concerns the ability to act through a different protocol rather than transfer to a different task distribution. The point-estimate recovery under fixed-runtime skin changes is consistent with an interface-mediated component that a pure capability-forgetting account does not predict. LLMs are known to be sensitive to surface features of prompts (Sclar et al., 2024; Mizrahi et al., 2023). Tool-use pretraining (Schick et al., 2023) fixes a tool vocabulary into the model, and tool-use benchmarks diagnose failures at scale (Soni, 2026). We show the agentic analogue: SFT can bind task-solving behavior to the specific identifier realization of the training harness. Enforcing agreement between two views of an input is classic in semi-supervised and robust training (Sohn et al., 2020; Liang et al., 2021), and invariance across environments is the goal of IRM-style objectives (Arjovsky et al., 2019). ARCHER is a token-level instance of this idea using cross-task shuffled views; the agreement target is per-token difficulty, while the shuffled construction supplies interface and packing diversity.

3 Native SFT Gains Can Reverse Under a Harness Change

3.1 Setup: Tasks, Harnesses, and Two Deltas

We consider a set of coding tasks $\mathcal{T}$ and a set of coding-agent harnesses $\mathcal{H}$. A harness decomposes into a runtime (the execution environment), a tool inventory (names, parameter schemas, result formats), a serializer and parser for the model's outputs, the system prompt, and the interaction budget; different harnesses may provide overlapping functionality while exposing it through different tool names, schemas, prompts, and message conventions (Ruan et al., 2026; Pan et al., 2026; Ning et al., 2026). Let $\mathcal{D}_h$ denote trajectories collected through harness h, M_0 the base model, and $M_h = \mathrm{SFT}(M_0, \mathcal{D}_h)$. With $P(M, h)$ the task success rate of model M deployed through harness h, we track

$$\Delta_{\mathrm{native}}(h) = P(M_h, h) - P(M_0, h), \tag{1}$$

$$\Delta_{\mathrm{transfer}}(h, h') = P(M_h, h') - P(M_0, h'), \quad h' \neq h. \tag{2}$$

We use the term *harness entanglement* to describe the phenomenon in which SFT preserves or improves performance on the training harness (nonnegative Δ_{native}) while selectively and asymmetrically degrading performance on other harnesses (strongly negative $\Delta_{\mathrm{transfer}}$). The definition is behavioral; our skin experiments (Section 4.3) separately identify an interface-mediated component of this degradation. Agentic SFT *can induce* this phenomenon – it does not always (Section 4.1). The formulation does not assume that skill and harness behavior are separable in the model; it is a lens for analyzing how the two become entangled through SFT.

All main experiments use Qwen3.5-9B with three training harnesses: Claude Code, OpenHands, and DeepSeek Harness. Training starts from an aligned pool of 1,341 coding tasks. A native single-harness corpus contains one teacher trajectory per task; the two-harness mixture contains 2,682 trajectories, one per task and harness. All trajectories are produced by DeepSeek v4 Flash, with temperature 1.0, top-p 0.95, and top-$k = -1$ at collection; token exposure is matched across arms as closely as the packed data permit. Each native corpus retains one teacher trajectory per aligned task. Evaluation is performed on held-out coding tasks from SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024). The principal SFT arms use matched token budgets, optimizer settings, and a 262,144-token packing limit; the paired arms additionally keep both members of each retained pair in the same pack.

Table 1: Primary cross-harness results on SWE-BENCH Verified (success rate, %), under the unified evaluation protocol. Entries are mean ± sample SD over three independent evaluations on the same task set. Bold marks the highest mean in each column within this table. CLAUDE CODE, OPENHANDS, DEEPSEEK HARNESS, and PI denote Claude Code, OpenHands, DeepSeek Harness, and Pi, respectively.

Training condition	CLAUDE CODE	OPENHANDS	DEEPSEEK HARNESS	PI
Base Qwen3.5-9B	50.6±0.8	47.0±1.9	50.6±0.6	49.9±0.9
SFT-CC	**51.4±1.7**	38.7±1.4	40.8±0.8	44.0±2.4
SFT-Mix	49.1±1.1	61.7±2.2	45.9±2.6	52.5±3.9
SFT-Mix-randomized	46.5±2.4	50.2±3.1	53.1±3.5	55.0±1.6
Content-preserving	49.8±2.5	60.7±3.1	50.9±2.2	53.8±2.2
ARCHER	51.3±1.8	**65.4±3.5**	**62.2±2.7**	**63.2±0.2**

DEEPSEEK HARNESS is held out from the training data of the principal transfer arms (SFT-CC, SFT-Mix, ARCHER) and therefore serves as the held-out deployment testbed; the separately trained native-DEEPSEEK HARNESS arm (SFT-DSH) serves as an oracle reference because its training data include DEEPSEEK HARNESS. The choice of DEEPSEEK HARNESS as the primary transfer harness reduces the confound of prior exposure to the target interface. CLAUDE CODE and OPENHANDS were public before the base model was released, whereas DEEPSEEK HARNESS was released only afterward; its specific interaction protocol was therefore unlikely to have appeared in the base model's pretraining data. DEEPSEEK HARNESS is also absent from all transfer-arm SFT data; we use it as the primary SFT-held-out transfer harness (Appendix J). We additionally evaluate PI, an independent held-out harness absent from the transfer-arm training data, as a secondary check that the observed transfer pattern is not specific to DEEPSEEK HARNESS. We use a task-disjoint split from the same SWE-BENCH distribution, not a repository-disjoint split, so the transfer result concerns interface changes on shared repositories. All main-table cells use one evaluation protocol (Appendix A.3). For each condition and harness, we conduct three independent evaluations on the same task set and report the mean and sample standard deviation. An episode is successful when its submitted patch passes the official SWE-BENCH Verified grader, including the required fail-to-pass and pass-to-pass tests. Task-level loss, overlap, and complexity analyses use separate single-run diagnostic traces under the same protocol.

The training matrix is organized around two controls. SFT-CC isolates the effect of collecting trajectories in one harness, while SFT-Mix asks whether exposing the student to two native interfaces is enough to prevent the loss. The randomized mixture changes identifier realization without pairing views of the same task. The content-preserving arm pairs each trajectory from the CLAUDE CODE +OPENHANDS pool with its identifier-renamed rendering, preserving task content. Shuffled-NC removes the consistency term from the shuffled construction, and Shuffled-random-mask keeps the construction but selects consistency positions without using target-token identity. ARCHER is the remaining arm: it uses the shuffled pairs and the identity-based position rule. These controls compare pairing constructions and consistency rules.

3.2 NATIVE GAIN, NON-NATIVE LOSS

We begin with SFT on native CLAUDE CODE trajectories (SFT-CC). Table 1 compares this model with the base model on all four harnesses, including held-out PI. The native controls appear in Table 2; the training interventions are discussed in Section 5.

The omitted token-level regularizers, masking control, paired-view R-Drop, and content-preserving weight variants are reported in Appendix B.

SFT-CC slightly raises native CLAUDE CODE success but falls below the base model on all three other harnesses, with the largest loss on DEEPSEEK HARNESS at 9.8 points. The base model's comparable scores across harnesses rule out a simple difference in benchmark difficulty. The same task pool therefore supports both native success and non-native failure, depending on the harness.

Native OPENHANDS and native DEEPSEEK HARNESS improve on the protocols represented in their own trajectories, while transcribed OPENHANDS does not match native OPENHANDS after the

Table 2: Key controls and attribution ablations on SWE-BENCH Verified (success rate, %), under the same unified evaluation protocol as Table 1. The upper block tests native harness and transcription controls; the lower block compares consistency variants within the shuffled construction. The native control rows and shuffled rows summarize three independent evaluations. Additional regularizer and weight-sensitivity results appear in Appendix B.

Training condition	CLAUDE CODE	OPENHANDS	DEEPSEEK HARNESS	PI
SFT-OH (transcribed)	40.5±1.5	44.7±3.4	19.6±1.8	21.5±0.7
SFT-OH (native)	55.2±0.6	66.4±4.0	38.6±2.8	57.6±2.6
SFT-DSH (same-harness oracle)	50.9±2.4	27.8±1.1	69.2±1.7	54.7±0.6
Shuffled-NC	50.6±0.9	62.8±4.0	56.1±0.9	60.3±0.8
Shuffled-random-mask	49.2±2.7	64.3±3.7	60.4±1.2	61.6±1.1

surface format is rewritten. The shuffled controls compare consistency variants within the shuffled construction.

3.3 LOST TASKS, NOT JUST LOWER SCORES

We also count base-solvable tasks that become failures after SFT. SFT-CC loses over one-third of those tasks on OPENHANDS and DEEPSEEK HARNESS, compared with 16.0% for native-OPENHANDS on its home harness. The content-preserving arm loses 21.8% on DEEPSEEK HARNESS, while transcribed-OPENHANDS loses 71.4%. The score gap therefore reflects task turnover, not only marginal changes in success rate. The denominator is the set of tasks solved by the base model under the same harness, so the statistic measures lost successes; full retention and overlap analyses appear in Appendix D.

This task-level view connects the aggregate score to the interaction traces. The observed traces illustrate two routes to failure: a model can emit an incompatible first call and stop early, or it can continue producing rejected calls and exhaust its interaction budget. The retention analysis therefore complements the protocol probe: the former measures which tasks disappear, while the latter identifies how the episode begins to fail. The appendix also compares the base-model episode length and tool-call count for retained and lost tasks. Lost tasks have longer base-model episodes and more tool calls in the reported comparisons, linking task turnover to the demands of sustained interaction without changing the task definitions or grader.

4 DIAGNOSING HARNESS ENTANGLEMENT

4.1 NATIVE CONTROLS

Native-OPENHANDS and native-DEEPSEEK HARNESS improve their home harnesses but fall below base on each other's harness. These opposite patterns show that the training interface matters; fine-tuning does not impose a uniform performance penalty. The native-DEEPSEEK HARNESS model also shows that SFT can improve the target score when its protocol appears during collection. Transcribed-OPENHANDS reaches 44.7% on OPENHANDS, below native-OPENHANDS at 66.4%. Conversion therefore does not reproduce native collection. The corpora also differ in source behavior, tool pacing, reasoning style, and unmatched conversion events (Appendix A.7). This comparison tests the two collection routes as a whole, rather than isolating surface syntax.

4.2 PROTOCOL MISMATCH IN TOOL CALLS

Invalid interactions provide a concrete signature of protocol mismatch. On held-out DEEPSEEK HARNESS, 21.9% of SFT-CC tool-call attempts are rejected, versus 5.3% for the content-preserving paired construction and 2.4% for the shuffled main arm (Figure 3 in the appendix; the distributions are in Appendix C.2). In the separate onset analysis, first rejections appear near the start of the transcribed-OPENHANDS and native-DEEPSEEK HARNESS traces on CLAUDE CODE (median parsed-marker index 1), but later for SFT-CC on DEEPSEEK HARNESS (median index 17). Models often request tools from the training harness or fabricate responses after rejection. The paired con-

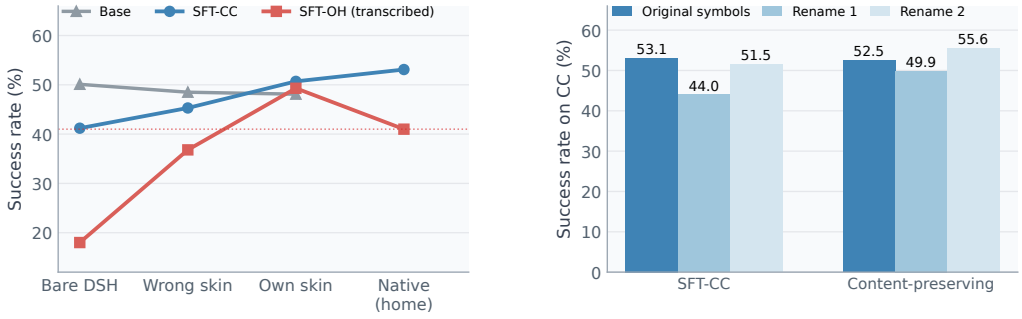


Figure 2: Two controlled interventions. **Left:** interface-skin intervention on a fixed DEEPSEEK HARNESS runtime. Within each model, the bare, CLAUDE CODE-skin, and OPENHANDS-skin cells use the same model, task IDs, and runtime, but generate separate stochastic episodes; only the outer interface configuration differs. The dotted line marks transcribed-OPENHANDS's home score as a cross-runtime reference. The fine-tuned single-harness models' own-skin point estimates exceed their bare-DEEPSEEK HARNESS point estimates. Base-model skin controls are included as a reference. **Right:** deployment-side tool-identity sensitivity under two deterministic identifier-renaming maps. SFT-CC has a 9.1-point drop under map 1, compared with 2.6 points for the content-preserving paired construction; map 2 changes the size and sign of the paired construction's change, showing that the effect depends on the particular rename map.

structions more often produce executable calls, whereas control-arm rejections on CLAUDE CODE occur early in the generated interaction. The parser definition and full analyses are in Appendix D.3.

The two failure shapes matter for interpretation. An early rejection indicates that the model selected a tool or message convention unavailable in the target harness. Repeated rejections indicate that the model continues a learned interaction loop after the runtime has already signaled a mismatch. Both are counted as invalid interactions, but only the second exposes persistence after feedback; the trace analysis keeps these cases separate.

4.3 CHANGE THE SKIN, KEEP THE RUNTIME

We hold the underlying held-out DEEPSEEK HARNESS runtime fixed and vary only its outer interface, using skins that match the training harnesses (Figure 2, left). Changing the outer interface raises transcribed-OPENHANDS from 18.0% to 49.3% and SFT-CC from 41.2% to 50.7% on the fixed runtime. The skin therefore recovers part of the loss. Full skin matrices and construction details appear in Appendix E.

The intervention preserves the repository setup, grader, runtime, task IDs, and model. Each skin jointly changes the system prompt, tool schemas, message history, and result wrappers; an adapter maps calls to the same runtime API. The recovery therefore concerns the interface configuration as a whole, including instruction content, formatting, and context length. It does not isolate familiar names or syntax. The separate identifier probe below targets lexical changes, while base-model skin evaluations provide a reference for the same bundled intervention (Appendix A.5).

The skin experiment changes the wrapper around the runtime. We next change the symbols inside that wrapper, then ask whether the base model already prefers one rendering before any trajectory SFT.

4.4 IDENTIFIER SENSITIVITY

We next vary only tool and parameter identifiers while keeping the runtime fixed. Under one deterministic rename map, SFT-CC drops by 9.1 points, whereas the content-preserving paired construction drops by 2.6 points. A second map changes the size and sign of the paired construction's change, showing that identifier sensitivity depends on the replacement pattern.

Both maps hold the model, task IDs, runtime, and evaluation budget fixed, testing identifier sensitivity under two replacement patterns.

The renaming probe preserves tool functionality, argument types, execution semantics, and the underlying runtime. It changes only the lexical symbols that the model must emit and parse. A larger drop for SFT-CC than for the content-preserving arm therefore indicates dependence on the learned symbol surface, while the different response to the second map shows that this dependence is pattern-specific rather than a single fixed penalty. Appendix A.4 gives map 1 and the training-side OPENHANDS control.

4.5 A BASE-MODEL PROBE FINDS NO STRONG CC/OH PRIOR

We ask whether the base model already favors one interface before trajectory SFT. The probe compares matched content rendered in CLAUDE CODE and OPENHANDS, reducing differences in task and continuation content. Across 40 pairs, the mean CLAUDE CODE–OPENHANDS cross-entropy difference is -0.0265 nats, and CLAUDE CODE has higher loss on half of the pairs (Appendix F). The scores evaluate recorded target tokens under teacher forcing and measure rendering preference. This balanced result provides a pre-SFT reference for the post-SFT skin and rename effects, which concern how the trained agent interacts with its deployment harness during execution.

5 TRAINING-SIDE MITIGATION

The skin diagnostics leave a training question: can the student depend less on the interface seen during collection? We compare corpus-level diversification (§5.1), token-level controls (§5.2), and paired consistency (§5.3); the shuffled-pair version transfers best.

5.1 MIXING AND IDENTIFIER RANDOMIZATION

The simplest intervention is ordinary SFT on the union of native CLAUDE CODE and OPENHANDS trajectories. SFT-Mix reaches 45.9% on DEEPSEEK HARNESS, above SFT-CC at 40.8% but below the base model at 50.6%; on PI it reaches 52.5%, above the base model's 49.9%. Mixing adds interface diversity without linking equivalent content across renderings. Adding the held-out harness to the training mixture improves its same-side score but costs OPENHANDS performance (Appendix H).

Randomization renders each trajectory with freshly sampled identifiers, a form of domain randomization (Tobin et al., 2017). The randomized mixture reaches 53.1% on DEEPSEEK HARNESS and 55.0% on PI, but its OPENHANDS score is 50.2% versus 61.7% for the unrandomized mixture (Table 1). It does not couple equivalent renderings (Appendix A.4). ARCHER (§5.3) instead uses cross-task shuffled pairs and a consistency objective.

Pi provides a useful secondary comparison because it is held out from the transfer-arm training pool. The ordinary mixture already exceeds the base model on Pi, so its 52.5% score does not represent the same failure as the 45.9% DSH score. Randomized identifiers raise both held-out scores relative to that mixture but reduce OPENHANDS performance; ARCHER improves both held-out harnesses again. We therefore treat Pi as a check on the direction of the effect, rather than as a duplicate of the primary DSH diagnosis.

5.2 TOKEN-LEVEL CONTROLS

The binding analysis motivates three token-level controls on complete tool-call blocks, including delimiters, tool names, and arguments. On these spans, Anchor-preserve penalizes target-token NLL above the frozen base model's NLL, discouraging deterioration in the likelihood of recorded calls. Anchor-cap penalizes NLL below the base reference, constraining additional confidence in the training protocol. Both retain ordinary SFT supervision on these blocks. Proto-mask excludes these targets from the SFT loss while retaining the full trajectory as input.

None restores cross-harness performance while retaining seen-harness performance; masking collapses tool use to zero. The masked model describes intended operations in prose and code blocks, makes no actual tool calls, and ends its episodes with empty diffs. This outcome motivates retaining

direct supervision of executable calls when adding an auxiliary transfer objective. A paired-view R-Drop control reaches 30.6% on DEEPSEEK HARNESS. R-Drop matches full predictive distributions across dropout views of the same trajectory, whereas ARCHER matches scalar target-token losses across shuffled views and changes the packing context. The comparison changes the data context as well as the loss term: R-Drop regularizes repeated views of one trajectory, while ARCHER couples losses from different tasks. Its result compares these training constructions, rather than isolating a failure of distribution matching. The objectives and diagnostic statistics are in Appendix A.6.

5.3 ACROSS-TASK RANDOMIZED CONSISTENCY

We turn the same idea into a training objective. We construct (x^A, x^B) by shuffling the second views across source tasks in the mixed training pool. Each branch retains its own rendered trajectory, with token loss $\ell_t^r = -\log p_\theta(y_t^r \mid y_{<t}^r)$. Let J_{AB} contain the jointly supervised positions within the shared prefix of length $\min(|x^A|, |x^B|)$, indexed from the start of each row. ARCHER compares losses where the target-token IDs agree:

$$M_{AB} = \{t \in J_{AB} : y_t^A = y_t^B\}. \tag{3}$$

For the pairs packed into a microbatch $\mathcal{P}$, let $S_\mathcal{P}$ be all supervised token positions and $N_\mathcal{P} = |S_\mathcal{P}|$. The objective is

$$\mathcal{L}_{\text{total}}(\mathcal{P}) = \frac{1}{N_\mathcal{P}} \left[\sum_{u \in S_\mathcal{P}} \ell_u + \lambda \sum_{(A,B)\in\mathcal{P}} \sum_{t \in M_{AB}} \left(\ell_t^A - \ell_t^B\right)^2 \right]. \tag{4}$$

The SFT and consistency sums share the same supervised-token denominator; consistency is not additionally divided by $|M_{AB}|$. Losses are averaged over microbatches, with gradients through both branches. The shared-prefix restriction applies only to consistency: SFT supervises the full retained rows, including tokens beyond that window. Padding and ignored targets are excluded by the supervision mask. Appendix G gives the packing and mask details.

The construction has three distinct steps. First, each source trajectory in the mixed training pool supplies an original and an identifier-renamed rendering, with the underlying trajectory content preserved. Second, the second rendering is permuted across source tasks before the two views are packed together, so co-occurrence supplies interface diversity without creating a semantic target for the paired task. Third, the loss gap is evaluated at the sites in M_{AB}. Shuffled-random-mask instead samples exactly $|M_{AB}|$ positions uniformly without replacement from J_{AB}, allowing unequal target IDs. This separates the source of diversity from the choice of consistency sites.

For a selected position, the two losses are computed against each view's own recorded continuation. Equal target IDs specify where to compare the scalar losses; their preceding contexts can come from different tasks. A shared punctuation or numeric token can therefore qualify without denoting a shared coding action. The ordinary SFT term continues to supervise the trajectory content outside the selected sites, so the consistency mask does not replace the task-learning signal. This distinction separates the objective from matching whole predictive distributions and explains why a random-position control is informative: it tests whether target identity adds value beyond coupling the difficulty of supervised tokens.

ARCHER, Shuffled-NC, and Shuffled-random-mask use identical packed training data: source rows, renamed views, cross-task permutation, pack boundaries, and SFT targets are shared, as are the optimizer schedules. Only consistency weight and mask selection differ. These controls isolate the loss choices within the shuffled construction. The comparison with the content-preserving arm also changes pairing, retained examples, and the consistency window.

At each selected position, consistency matches the losses of the two continuations without forcing their tokens, tool calls, or actions to agree. This keeps the term applicable when the two tasks differ and avoids treating surface-level equality as a semantic correspondence. The resulting transfer behavior is therefore an empirical test of whether this weak local constraint is enough to reduce dependence on a particular harness. We call this approach **ARCHER** (**A**cross-task **R**andomized **C**onsistency for **H**arness **E**ntanglement **R**eduction). The earlier same-trajectory construction is retained as a content-preserving ablation.

We use the default λ=1, fixed before any evaluation of the shuffled construction. The retrospective $\lambda \in \{0.5, 1, 2\}$ sensitivity analysis uses the content-preserving construction and did not determine ARCHER's λ. PI was introduced after the training configurations were fixed and was not used for method or hyperparameter selection; we evaluate the same ARCHER configuration on both held-out harnesses.

5.4 WHAT DRIVES TRANSFER?

On DEEPSEEK HARNESS/PI, the shuffled-NC, random-position, and ARCHER scores are 56.1/60.3, 60.4/61.6, and 62.2/63.2. Both consistency variants exceed Shuffled-NC; ARCHER's means are 1.8 and 1.6 points above the random-position control. The PI result extends this ordering to the additional held-out interface. The full means appear in Tables 1 and 2; content-preserving sweeps are in Appendix G.

The reported content-preserving diagnostic variants remain below ARCHER. The fresh-rename teacher-forced NLL probe is nearly unchanged between the content-preserving no-consistency and consistency arms. The shuffled variants transfer better despite pairing different tasks: matched task identity is not required for the gains observed here. Detailed scores appear in Appendix G.

The content-preserving no-consistency control tests whether paired data alone account for the improvement. It retains the same source–renamed pairs and packing schedule, including pairs that receive SFT but no consistency loss. Adding consistency raises mean DSH success from 49.1% to 50.9%, while Pi falls from 55.6% to 53.8%. The content-preserving construction therefore shows no uniform transfer benefit from consistency. The shuffled controls show a different pattern: both consistency variants improve the mean on each held-out harness, supporting consistency without matched task identity.

Two further controls examine what the paired construction needs. Restricting paired training to one harness yields DSH point estimates of 41.2% for CC-only and 33.7% for OH-only. Thus pairing with identifier variants inside one harness does not reproduce the transfer of the mixed construction. A separate high-coverage construction increases matched-supervision coverage from 17.5% to 88%, but its diagnostic point estimates remain below ARCHER. Reducing the consistency weight or normalizing by the number of matched positions improves some high-coverage variants. Neither adjustment reproduces the shuffled construction's transfer, so these results motivate examining pairing diversity alongside the number of constrained positions. This comparison changes the examples and total consistency pressure together; it tests the construction rather than coverage in isolation. The full auxiliary results are in Appendix G.

These comparisons support loss consistency within the shuffled construction, but do not isolate cross-task co-occurrence from packing, retention, or comparison-window effects. Both position rules improve the held-out means, so exact token-ID agreement is not essential to the observed gain. The ID-based mask may also select different token types and locations; its small mean advantage does not by itself establish a benefit from identity alone. Equal target IDs define a scalar-loss comparison, not a correspondence between coding actions.

6 CONCLUSION

Trajectory SFT can improve native execution while reducing transfer. The native controls show that the direction of transfer depends on the training harness, and deterministic transcription does not reproduce native collection. Fixed-runtime skins and identifier changes locate an interface-mediated component of the loss; task retention and interaction traces show how that loss appears in execution. ARCHER combines cross-task shuffled packing with position-level loss consistency. On DSH it overcomes the degradation that persists under ordinary mixed SFT; on Pi it improves beyond a mixture that already exceeds the base model. Within the shuffled construction, both consistency variants exceed the no-consistency control, and ARCHER has the highest held-out means. These results motivate evaluating distilled coding agents across interfaces and including interface diversity in training. Native success alone does not show whether the learned behavior will survive a change of harness.

Reproducibility Statement. Our experiments use Qwen3.5-9B and an aligned training pool of 1,341 SWE-BENCH tasks (Jimenez et al., 2024). Single-harness arms use one trajectory per task, while the unpaired two-harness mixture uses one trajectory per task and harness; ARCHER additionally constructs cross-task shuffled paired views. We evaluate on SWE-BENCH Verified (OpenAI, 2024) with four evaluation harnesses (CLAUDE CODE, OPENHANDS, DEEPSEEK HARNESS, and PI) under a single unified evaluation protocol. The main transfer and mitigation arms exclude DEEPSEEK HARNESS training data; the DEEPSEEK HARNESS-native oracle and three-harness mixture are explicitly reported as same-harness references. Training recipes, rendering and renaming procedures, skin construction, and detector definitions are specified in the appendices.

Ethics Statement. This work studies coding-agent behavior and does not involve human subjects or decision-making about individuals. The primary ethical consideration is the potential resource consumption caused by pathological agent interaction, such as repeated invalid tool calls. Our analysis exposes these failure modes to facilitate safer evaluation and deployment of autonomous coding systems.

AI Use Statement. Generative AI tools were used to assist with manuscript organization, language editing, and LaTeX preparation. The authors remain responsible for the experimental design, implementation, verification of results, interpretation of findings, and final contents of the submission.

REFERENCES

Agentica Project. DeepSWE: Training a State-of-the-Art Coding Agent from Scratch by Scaling RL. `https://pretty-radio-b75.notion.site/DeepSWE`, 2025.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward Language Agent Fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Weizheng Gu, Chengze Li, Zhuohao Yu, Mengyuan Sun, Zhibang Yang, Wei Wang, Hongrui Jia, Shikun Zhang, and Wei Ye. What Do Agents Learn from Trajectory-SFT: Semantics or Interfaces? *arXiv preprint arXiv:2602.01611*, 2026.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The False Promise of Imitating Proprietary LLMs. In *International Conference on Learning Representations*, 2024.

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 *International Conference on Learning Representations*, 2024.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling LLM Agent into Small Models with Retrieval and Code Tools. *arXiv preprint arXiv:2505.17612*, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Han Li, Zhemin Fang, Rili Feng, Yingqi Zhao, Jiaheng Liu, Pengfei Gao, He Ye, Dayi Lin, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. LoopsBench: From Harness Engineering to Loop Engineering in Coding Agent Evaluation. *arXiv preprint arXiv:2608.00267*, 2026.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. R-Drop: Regularized Dropout for Neural Networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 10890–10905, 2021.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of What Art? A Call for Multi-Prompt LLM Evaluation. *arXiv preprint arXiv:2401.00595*, 2023.

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, Lingjie Chen, Yanjun Zhao, Ke Yang, Bingxuan Li, Cheng Qian, Gaotang Li, Xiao Lin, Zhichen Zeng, Ruizhong Qiu, Sirui Chen, Yifan Sun, Xiyuan Yang, Ruida Wang, Rui Pan, Chenyuan Yang, Dylan Zhang, Liri Fang, Zikun Cui, Yang Cao, Pan Chen, Dorothy Sun, Ren Chen, Mahesh Srinivasan, Nipun Mathur, Yinglong Xia, Hong Li, Hong Yan, Pan Lu, Lingming Zhang, Tong Zhang, Hanghang Tong, and Jingrui He. Code as Agent Harness. *arXiv preprint arXiv:2605.18747*, 2026.

OpenAI. Introducing SWE-bench Verified. `https://openai.com/index/introducing-swe-bench-verified/`, 2024.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym. *arXiv preprint arXiv:2412.21139*, 2024.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. Natural-Language Agent Harnesses. *arXiv preprint arXiv:2603.25723*, 2026.

Junhao Ruan, Yuan Ge, Bei Li, Yongjing Yin, Yuchun Fan, Xin Chen, Jingang Wang, Chenglong Wang, Jingbo Zhu, and Tong Xiao. ToFu: A White-Box, Token-Efficient Agent Harness for Researchers. *arXiv preprint arXiv:2607.11423*, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design. In *International Conference on Learning Representations*, 2024.

Shuai Shao, Kangning Zhang, Qingyao Li, Shijian Wang, Hao Wang, Wenxiang Jiao, Yuan Lu, Yi Guo, Weiwen Liu, and Weinan Zhang. Harness-R1: Learning to Edit Executable Runtime Harnesses from Agent Failure Trajectories. *arXiv preprint arXiv:2608.02276*, 2026.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *Advances in Neural Information Processing Systems*, volume 33, pp. 596–608, 2020.

Harsh Soni. ToolFailBench: Diagnosing Tool-Use Failures in LLM Agents. *arXiv preprint arXiv:2607.04686*, 2026.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. doi: 10.1109/IROS.2017.8202133.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents. *arXiv preprint arXiv:2407.16741*, 2024.

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, pp. 50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling Generalized Agent Abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

A Experimental Details

A.1 Evaluation Task Set

All task-success evaluations use SWE-bench Verified (OpenAI, 2024), which contains 500 coding tasks. We evaluate on 495 tasks: five tasks were excluded because their internal evaluation environments were unavailable during construction. The excluded identifiers are `django__django-10097`, `django__django-16502`, `django__django-7530`, `psf__requests-1724`, and `sympy__sympy-24539`. The same five tasks are excluded for all harnesses under the unified protocol.

A.2 Models, Data, and Training

All main experiments use Qwen3.5-9B as the base language model. Each native single-harness corpus contains 1,341 retained trajectories from the same aligned task pool; the two-harness mixture contains 2,682 trajectories and uses fewer epochs to match the training budget. All arms share a single teacher, DeepSeek v4 Flash, which produced the native Claude Code, OpenHands, and DeepSeek Harness trajectories with temperature 1.0, top-p 0.95, and top-$k = -1$. The transcribed arms (SFT-OH (transcribed), OH→CC (transcribed)) re-render the source trajectories in another interface, subject to the transcription caveats in Appendix A.7; the renamed and randomized arms change only identifiers, and the regularized arms reuse SFT-CC or SFT-Mix data with no new teacher. The student is trained with standard SFT in all cases. Main-table evaluation is conducted on held-out tasks from SWE-bench Verified through the native runtime of each harness. DeepSeek Harness is held out from the principal transfer arms and serves as a training-held-out deployment environment; SFT-DSH and the three-harness mixture are same-harness references.

We also made exploratory attempts with smaller student models to assess whether the phenomenon could be measured outside Qwen3.5-9B. Gemma 4 E4B produced zero successful episodes across the tested harnesses under the same evaluation protocol, so its coding capability was insufficient for observing cross-harness differences. Qwen3.5-2B produced only single-digit success rates, which likewise left too little headroom for a meaningful comparison of the transfer conditions. We therefore do not include either model in the quantitative claims or tables of this paper.

All principal SFT conditions use matched token budgets, optimizer settings, and a 262,144-token packing limit. Single-harness arms expose 1,341 rows for four epochs; SFT-Mix exposes its 2,682 rows for two epochs, giving 5,364 row presentations in either case. The content-preserving paired construction starts from the same raw 2,682-row Claude Code +OpenHands corpus and pairs each source row with one fixed renamed rendering. The ARCHER construction cross-associates the second views across source tasks before atomic packing; its no-consistency and random-position variants use the same shuffled packing and schedule. The content-preserving paired arms retain 1,827 distinct pairs whose combined rendered length fits the context limit, and the consistency term further skips 165 unequal-length pairs, leaving 1,662 equal-length eligible pairs. These counts precede the two training passes encoded in the packs (Appendix G). For unequal-length shuffled pairs, the consistency window is the shared prefix; SFT uses both full retained rows. The 17.5% coverage statistic below applies only to the content-preserving audit; we do not use it as a coverage estimate for ARCHER. SFT-Mix instead uses ordinary per-sequence packing, so it is matched in token budget and schedule but not necessarily in the raw trajectories that survive packing. Random names are

sampled once per trajectory before training and remain fixed across epochs; shuffled views are fixed before packing. The shuffled pairing permutation and pair manifest are fixed before training. One shuffled pair is an accidental same-task Claude Code/OpenHands pairing; all remaining pairings are cross-task.

Training uses 262,144-token packed sequences, AdamW, learning rate 10^{-4}, global batch size 32, and bf16. SFT-CC and SFT-OH (transcribed) run for 89 optimizer steps, SFT-Mix and the paired arms for 90, SFT-OH (native) for 91, and SFT-DSH for 55. Evaluation decodes with temperature 1.0, top-p 0.95, top-$k = -1$, a 262,144-token context limit, a 65,536-token generation limit, a 4,800-second agent timeout, and a 9,600-second external guard timeout, with no fixed turn limit. Each task uses the same fixed prebuilt 4-core, 16-GiB sandbox template and shared mount configuration across arms. Within each comparison, the harness and sandbox configurations are held fixed. The training task pool and the 495 evaluation tasks are disjoint at the task level (Appendix A.1); this is a task-ID split with repository overlap. The training set covers 11 repositories, the evaluation set 12, and their intersection is 11 repositories.

Additional optimization details. The recorded recipe specifies the model and teacher identities, retained data counts, packing and step budgets, optimizer family, learning rate, batch size, precision, and decoding parameters above. The canonical ARCHER configuration uses AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, decoupled weight decay 0.01, cosine decay to zero, a 10% warmup fraction, micro-batch size 1 with no additional gradient accumulation, and gradient clipping at 1.0. It uses dataset seed 1234 and Megatron RNG seed 5678; decoding uses the sampling parameters above without a fixed decode seed. The evaluation software and benchmark snapshots use pinned revisions of the harness and SWE-BENCH Verified evaluation pools. Reproduction uses the retained trajectories and immutable pairing manifest rather than rerunning teacher collection.

A.3 Experimental Conditions

All main-table success-rate results use a single unified evaluation protocol: a 4,800-second agent timeout, a 9,600-second external guard timeout, no fixed turn limit, one fixed prebuilt 4-core, 16-GiB sandbox template per task, and a shared mount plane. We conduct three independent evaluations for each main-table condition and harness. Within each evaluation run, every condition–harness–task tuple contributes exactly one retained stochastic episode. If an infrastructure or vendor failure prevents a usable episode on the first attempt, only that failed task is rerun; retained records are deduplicated by task identifier before scoring, so these operational replacements are not repeated evaluation samples.

Table 3 lists the principal training conditions; the position, coverage, strength, and single-harness variants are defined with their corresponding ablations in Appendix G.

A.4 Tool-Symbol Renaming

SFT-CC-renamed and SFT-OH-renamed replace selected tool and parameter identifiers with fixed alternate identifiers while preserving tool functionality, schema structure and types, execution semantics, the number of tools, and the underlying runtime behavior. Table 4 lists these training-side mappings; the Claude Code map also defines deployment-side map 1 in Figure 2. Natural-language content is unchanged. The renamed models are evaluated on the original interface, so the experiment tests lexical dependence on tool symbols while holding the execution environment fixed within each comparison. Evaluated off its home interface, the renamed arms have lower cross-harness point estimates: SFT-CC-renamed scores 26.5% on OpenHands and 27.9% on DeepSeek Harness, and SFT-OH-renamed scores 3.2% on Claude Code and 14.1% on DeepSeek Harness. These observed point estimates are low under the two non-home interfaces.

Protocol-randomized training arm. We apply per-trajectory identifier randomization to the mixed Claude Code +OpenHands corpus. The arm improves held-out transfer relative to the unrandomized mixture, but its OpenHands mean falls to 50.2% from 61.7%. It remains below ARCHER on both held-out harnesses. This arm uses ordinary SFT without pairing or a cross-view consistency term; all numbers use the unified protocol.

Table 3: Summary of experimental conditions.

Condition	Training trajectories	Purpose
Base	None	Base-model control
SFT-CC	Native CC	Single-harness SFT
SFT-OH (transcribed)	CC content transcribed to OH	Single-harness SFT
SFT-Mix	Native CC + OH	Cross-harness SFT
SFT-OH (native)	Native OH	Native OH control
SFT-DSH	Native DSH	Native DSH control
SFT-CC-renamed	SFT-CC data + renamed tools	Symbol intervention
SFT-OH-renamed	Native OH data + renamed identifiers	Symbol intervention
OH→CC (transcribed)	OH content transcribed to CC	Transcription control
SFT-Mix-randomized	SFT-Mix + fresh identifiers per trajectory	Symbol decorrelation
Anchor-preserve	SFT-CC data + penalty: protocol NLL $>$ base	Regularized SFT
Anchor-cap	SFT-CC data + penalty: protocol NLL $<$ base	Regularized SFT
Proto-mask	SFT-CC data, protocol spans masked from loss	Regularized SFT
Content-preserving-NC	Same-trajectory paired renderings, no consistency	Data-matched control
Content-preserving	Same-trajectory paired renderings + consistency	Content-preserving ablation
Shuffled-NC	Cross-task shuffled pairs, no consistency	Packing control
Shuffled-random-mask	Shuffled pairs + random-position consistency	Regularized ablation
ARCHER	Shuffled pairs + ID-matched consistency	Main method
Three-harness mix	Native CC + OH + DSH	Full-mixture reference (§H)

A.5 Interface-Skin Construction

The skin intervention separates the execution runtime from the surface protocol presented to the model. The **bare** condition exposes the native DeepSeek Harness interaction format. The **own-skin** condition presents the same runtime using the system prompt, tool schemas, message wrapping, and result conventions of the harness the model was trained on. The **wrong-skin** condition uses the conventions of the other training harness. A skin-specific adapter performs the translation: it renders the selected prompt, schemas, message history, and result envelopes to the model, then maps the model's tool call back to the fixed DeepSeek Harness runtime API. Thus the runtime, tool implementations and capabilities, accepted parameter semantics, task set, sandbox image, runtime error text, and per-episode context and time budgets remain fixed; only the outer serialization and its adapter change. Each skin cell is a separate stochastic evaluation: the model and task IDs are shared across cells, but the decoded episodes are not.

A.6 Protocol-Span Regularization

Protocol positions $P(x)$ are the token spans of tool-call blocks, detected on the fly by scanning the label sequence for the token-ID subsequences of the opening and closing `<tool_call>` delimiters. The delimiters and all positions between them are marked as protocol positions. An unclosed block is marked through the end of the sequence; protocol-like strings in ordinary text are marked as well. $\mathrm{NLL}_{\mathrm{base}}$ is precomputed once over the packed corpus with the frozen base model and stored as an auxiliary channel alongside the token ids, so training pays no reference forward pass. For the content-preserving arm, the second rendering of each trajectory replaces tool and parameter identifiers with fresh random strings when length-preserving tokenizations exist, and the two renderings are packed into the same sequence group; the consistency mask M is computed dynamically at matched token positions (Appendix G).

Table 4: Fixed identifier mappings for the training-side renaming arms. The CLAUDE CODE mapping is also used as deployment-side map 1.

Arm	Type	Original	Replacement
CC	Tool	`Read`	`ToolAlpha`
	Tool	`Edit`	`ToolBeta`
	Tool	`Write`	`ToolGamma`
	Tool	`Bash`	`ToolDelta`
	Parameter	`file_path`	`param_path`
	Parameter	`offset`	`param_offset`
	Parameter	`limit`	`param_limit`
	Parameter	`old_string`	`param_old`
	Parameter	`new_string`	`param_new`
	Parameter	`replace_all`	`param_all`
	Parameter	`command`	`param_cmd`
	Parameter	`timeout`	`param_timeout`
	Parameter	`description`	`param_desc`
OH	Tool	`terminal`	`ToolAlpha`
	Tool	`file_editor`	`ToolBeta`
	Tool	`think`	`ToolGamma`
	Tool	`finish`	`ToolDelta`
	Tool	`task_tracker`	`ToolEpsilon`
	Parameter	`security_risk`	`param_risk`
	Parameter	`command`	`param_cmd`
	Parameter	`summary`	`param_summary`
	Parameter	`timeout`	`param_timeout`
	Parameter	`path`	`param_path`
	Parameter	`view_range`	`param_range`
	Parameter	`new_str`	`param_new`
	Parameter	`old_str`	`param_old`
	Parameter	`thought`	`param_thought`
	Parameter	`message`	`param_msg`
	Parameter	`file_text`	`param_text`
Unchanged OH fields: `is_input`, `reset`, `security_rule`, `_raw`, `security_rank`.			

Per-task diagnostics for the preserve arm. The one cell where the anchor clearly hurts is OPENHANDS. In an auxiliary standard-budget evaluation of the same unanchored and Anchor-preserve models (one retained stochastic episode per task, not averaged with Table 6), the corresponding cells read 38.0% and 20.0%, respectively. Among the 120 tasks the un-anchored model solves and the anchored model loses (against 31 in the reverse direction), 72% fail with a fully functional interaction – five or more executed calls, no rejections, no timeout – 22% time out, and only 5% show rejection loops. Most lost tasks in this diagnostic fail despite sustained valid interaction, showing that task-solving degradation can occur without the rejection loops described in Section 4.2.

Per-token diagnostics for the cap arm. The base model assigns near-zero loss to the bulk of both harnesses' protocol spans (median NLL $\approx 10^{-4}$). At these positions, the cap penalty is bounded by the small base-model NLL; larger penalties are possible where the base model assigns higher NLL. Inspection of the OPENHANDS episodes finds behavior consistent with this pressure: the policy invokes tools with fabricated names (`<function=Read>`, `<function=view>`) that do not exist in the runtime, receives rejection messages, and in some cases self-corrects but at a cost to the solve rate.

Content-preserving token fit. Under held-out DEEPSEEK HARNESS, the content-preserving paired construction's mean content-token NLL is 0.245, below the unpaired mixture's 0.310, while its protocol-token NLL stays near the base model's level. These averages show no content-token NLL inflation in this comparison.

Masked-arm behavior. The protocol-masked policy narrates the actions it would take – "let me explore the codebase," followed by a shell command inside a markdown code block – and never invokes a tool, ending each episode with an empty diff. The loss directly supervises the surrounding text while excluding tool-call targets, consistent with the observed shift from executable calls to descriptions of actions.

A.7 TRANSCRIPTION CENSUS

We census the dumps that produced the native and transcribed arms. Table 5 compares all four source/rendered trajectory groups on per-episode action counts and tool mix.

Table 5: Transcription census. Counts summarize source and rendered trajectories; conversion can change event boundaries.

Source	Median calls/episode	Tool mix (top 3)
CC native	59	Bash 71k / Read 18k / Edit 5k
CC → OH transcribed	58	terminal 71k / file_editor 23k / finish 1k
OH native	65	terminal 90k / file_editor 12k / think 2k
OH → CC transcribed	63	Bash 90k / Read 7k / Edit 5k

For the CLAUDE CODE →OPENHANDS corpus used to train SFT-OH (transcribed), a per-record static validator checks visible and reasoning text, argument values, target tools, and message structure for conversion caveats. It reports no flag for 1,113 of 1,341 records. All 1,341 records are retained; the remaining 228 have a flagged source-side tool event without a direct target mapping and/or an associated tool-result omission. The validator covers only CLAUDE CODE →OPENHANDS; the reverse conversion uses a separate deterministic path.

The transcription itself is a deterministic field and schema conversion; no LLM rewriting is used. Explicit tool mappings convert Read, Edit, Write, and Bash into the corresponding OpenHands file-editor and terminal operations, with field-level conversions such as `file_path`→`path`, `old_string`→`old_str`, and `offset/limit`→`view_range`. Required target-side fields are added with their documented defaults, reasoning channels and inline `<think>` blocks are converted between the two message formats, and tool results are rewrapped using the target harness convention. Unmatched events are retained in the audit statistics rather than silently discarded; the reverse direction uses a separate deterministic conversion path.

Confounds. The SFT-OH (transcribed)/SFT-OH (native) comparison in Section 4.1 bundles several factors: the source behavioral distribution, native idioms (`think` appears in 46% of native OPENHANDS records vs. 0% after transcription), tool pacing, and reasoning style. These arms test whether converted source trajectories reproduce gains from native collection; they do not isolate syntax from source behavior or conversion effects.

B FULL DIAGNOSTIC ABLATIONS

Table 6 reports auxiliary regularizers, masking controls, paired-view controls, and content-preserving weight variants under the unified evaluation protocol.

Table 6: Full diagnostic ablations on SWE-BENCH Verified (success rate, %). Entries are mean ± sample SD over three independent evaluations.

Training condition	CLAUDE CODE	OPENHANDS	DEEPSEEK HARNESS	PI
Anchor-preserve	51.8±2.1	27.4±2.8	47.5±2.2	45.8±1.8
Anchor-cap	48.1±2.8	23.9±1.9	41.0±1.1	21.3±2.3
Proto-mask	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Content-preserving-NC	50.8±1.4	64.9±3.6	49.1±3.0	55.6±0.8
Content-preserving (λ=2)	53.1±2.6	61.6±1.2	49.8±1.6	54.6±3.4
Content-preserving (λ=0.5)	50.2±2.5	60.1±1.8	53.2±1.8	53.7±0.7
R-Drop (paired views)	41.2±1.1	48.3±1.8	30.6±2.1	29.9±0.3

The token-level normalized-variance regularizer (nllvar) is reported as a separate auxiliary arm below.

Table 7: Nllvar diagnostic evaluation (success rate, %): mean ± sample SD over three independent evaluations.

Condition	CLAUDE CODE	OPENHANDS	DEEPSEEK HARNESS	PI
nllvar	48.8±1.2	61.8±2.3	36.5±4.1	45.4±1.6

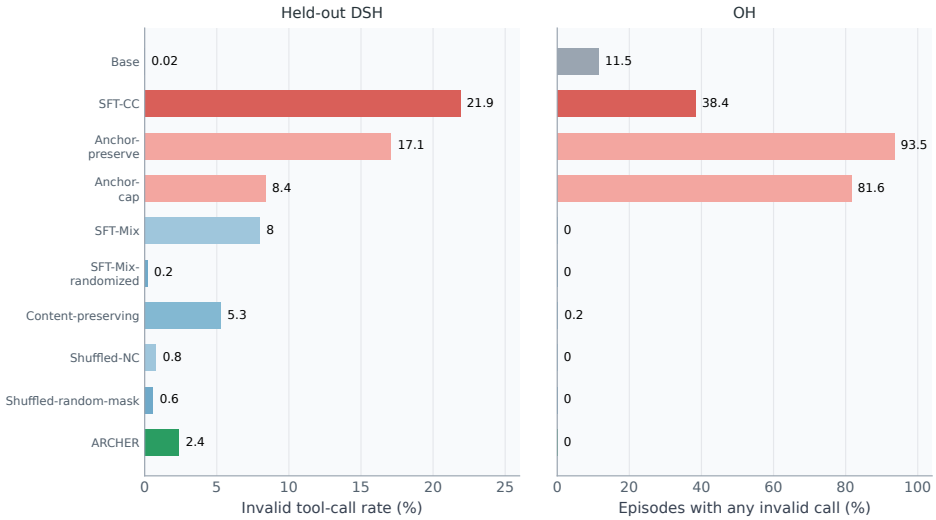


Figure 3: Invalid-interaction rates across training-side arms on the unified protocol. **Left:** invalid tool-call rate on held-out DEEPSEEK HARNESS. **Right:** share of OPENHANDS episodes containing at least one invalid call. OPENHANDS rejections are detected from the runtime's explicit rejection string (`Tool '<name>' not found`). The protocol-masked arm makes zero tool calls (collapse rather than invalid interaction) and is omitted. The content-preserving paired construction and the three shuffled-pair arms use the same probe; ARCHER is the main method among the shuffled arms. The rates use one diagnostic episode per task for each training-side arm.

C INTERACTION STATISTICS

Table 8 reports tool-call volume and execution rate for selected conditions, measured on the unified protocol. Calls/episode is the arithmetic mean across the 495 task-level episodes in that cell, with one retained episode per task; it is not an average over repeated decodes. Execution rate is the fraction of tool-call attempts that result in an actual tool execution. This differs from counting whether a tool-call message was generated, which matters for cross-harness evaluation: an SFT model can generate a syntactically valid call to a tool that does not exist in the target environment.

Table 8: Selected interaction statistics.

Condition	Calls/Episode	Execution Rate
Base × CLAUDE CODE	3.1	0.982
SFT-CC × CLAUDE CODE	3.4	1.000
SFT-OH (transcribed) × OPENHANDS	53.2	1.000
SFT-DSH × DEEPSEEK HARNESS	99.5	1.000
SFT-OH (native) × CLAUDE CODE	4.4	0.589
SFT-CC × DEEPSEEK HARNESS	184.6	0.615
SFT-OH (transcribed) × DEEPSEEK HARNESS	209.1	0.385

The home cells (above the rule) have execution rates at or near one. The cross-harness cells show two distinct failure shapes. SFT-OH (native) on CLAUDE CODE makes few calls and barely half execute: the model emits its home protocol, is rejected, and stops. SFT-OH (transcribed) on DEEPSEEK

HARNESS makes over 200 calls per episode with a 39% execution rate: the model keeps emitting rejected calls and, as described below, begins writing the observations it expected to receive.

C.1 CROSS-HARNESS REJECTION RATES

Table 9 reports the episode-level rejection rates for the single-harness and renamed arms across mismatched harnesses, measured on the unified protocol. Home cells are at or near zero; one home row is shown for calibration. In one `django` task a CLAUDE CODE-trained model generated runtime-rejected calls beginning with an attempt to invoke a `terminal` tool. In one bare-DEEPSEEK HARNESS evaluation the model produced 188 self-authored tool responses without executing a real tool.

Table 9: Harness-incompatible interaction behavior, unified protocol. Values are percentages of evaluation episodes containing at least one environment-rejected tool call ("invalid"), and episodes where the same tool is rejected at least three times ("persistent"). All three harnesses' rejection strings are known and detected (CLAUDE CODE: `No such tool available`; OPENHANDS: `Tool '<name>' not found`; DEEPSEEK HARNESS: `unknown tool`), so rates are comparable across the table.

Condition	Invalid-tool episodes	Persistent invalid
SFT-OH (transcribed) × CLAUDE CODE	53.1	21.8
SFT-OH (native) × CLAUDE CODE	49.3	27.3
SFT-DSH × CLAUDE CODE	22.8	11.5
SFT-CC-renamed × CLAUDE CODE	19.0	8.5
SFT-OH (transcribed) × DEEPSEEK HARNESS	99.4	98.0
SFT-OH (native) × DEEPSEEK HARNESS	80.6	65.7
SFT-OH-renamed × DEEPSEEK HARNESS	78.8	56.0
SFT-CC × DEEPSEEK HARNESS	74.1	65.7
SFT-DSH × DEEPSEEK HARNESS (home)	0.2	0.0

C.2 INVALID-INTERACTION DISTRIBUTIONS FOR THE TRAINING-SIDE ARMS

Table 10 gives selected-arm invalid-interaction summaries for the analysis in Section 4.2, measured on the unified evaluation protocol. On OPENHANDS, invalid calls are detected from the runtime's explicit rejection string (`Tool '<name>' not found. Available: [...]`); OPENHANDS tool calls are double-wrapped (`<tool_call><function=..>`), and both layers are merged before counting. Missing rejection fields are never treated as successful interaction.

Home cells (CLAUDE CODE) of all arms are at or near zero invalid calls, providing a calibration baseline. Two details are illustrative. The most frequently rejected identifier for SFT-CC on DEEPSEEK HARNESS is `replace_all` (15,328 rejections): the model promotes a *parameter name* of its home `Edit` tool to a function name. The base model's 82 invalid calls on OPENHANDS are genuine base-model errors (invoking the `file_editor` command `str_replace` as a tool name), not detector false positives. The content-preserving paired construction on OPENHANDS makes 35,344 tool calls, of which exactly one is rejected.

C.3 SELF-AUTHORED TOOL RESPONSES

The most severe failure shape, discussed in Section 4.2, occurs when the model produces text that imitates the structure of tool outputs without any corresponding runtime observation (the bare-DEEPSEEK HARNESS case with 188 self-authored responses). The model expects a particular interaction protocol and continues the learned sequence after the environment stops providing the expected response.

Table 10: Invalid-interaction distributions on DEEPSEEK HARNESS (top) and OPENHANDS (bottom). "Invalid %" is the number of environment-rejected calls divided by all tool-call attempts. "Any-invalid ep %" is the share of episodes with at least one rejected call, and "Persistent" is the share of episodes in which the same tool is rejected at least three times. Median/p95/max are per-episode invalid-call counts over all episodes, including episodes with zero invalid calls. The protocol-masked arm makes zero tool calls and is omitted.

Arm × DEEPSEEK HARNESS	**Invalid %**	**Persistent ep %**	**Median**	**p95**	**Max**
Base	0.02	0	0	0	1
SFT-CC	21.9	65.7	14	327	1406
Anchor-preserve	17.1	37.0	1	208	1221
Anchor-cap	8.4	41.2	1	112	299
SFT-Mix	8.0	26.1	0	87	1206
Content-preserving	5.3	23.2	0	29	1210

Arm × OPENHANDS	**Invalid %**	**Any-invalid ep %**	**Persistent ep %**
Base	0.2	11.5	0.8
SFT-CC	0.8	38.4	3.0
SFT-Mix	0	0	0
Anchor-preserve	4.3	93.5	10.9
Anchor-cap	3.1	81.6	10.3
Content-preserving	≈0	0.2	0

D ADDITIONAL TASK-LEVEL RESULTS

D.1 SOLVED-TASK RETENTION AND OVERLAP

Table 11 reports the conditional solved-task loss rate defined in Section 3.3 (denominator: tasks solved by the base model under the same harness) and the Jaccard overlap $|S(M_0) \cap S(M_h)|/|S(M_0) \cup S(M_h)|$ between solved-task sets, computed on the unified protocol. SFT-OH (native)'s 16.0% loss on its own OPENHANDS home gives a baseline for solved-set turnover after home-harness fine-tuning. The content-preserving paired construction loses 21.8% on held-out DEEPSEEK HARNESS, versus 71.4% for the transcribed arm on the same harness.

Table 11: Task-level solved-task retention relative to the base model on the unified protocol, using one diagnostic episode per model and task. "Loss" is the fraction of base-solvable tasks failed by the SFT model under the same harness; each cell shows Loss / Jaccard. The protocol-masked arm loses 100% (Jaccard 0.000) on all three listed harnesses and is omitted.

SFT condition	**CLAUDE CODE**	**OPENHANDS**	**DEEPSEEK HARNESS**
SFT-CC	21.1% / 0.627	36.0% / 0.531	35.5% / 0.548
SFT-OH (transcribed)	38.2% / 0.536	31.1% / 0.568	71.4% / 0.267
SFT-Mix	25.9% / 0.608	12.0% / 0.545	36.3% / 0.506
SFT-OH (native)	20.7% / 0.607	16.0% / 0.553	39.9% / 0.498
SFT-DSH	26.7% / 0.553	51.6% / 0.421	10.9% / 0.614
SFT-CC-renamed	35.1% / 0.558	–	–
SFT-OH-renamed	–	60.4% / 0.325	–
Anchor-preserve	25.1% / 0.595	59.1% / 0.355	27.4% / 0.619
Anchor-cap	29.1% / 0.597	60.9% / 0.358	35.1% / 0.555
Content-preserving	22.3% / 0.617	11.1% / 0.560	21.8% / 0.586

D.2 TASK COMPLEXITY AND CAPABILITY LOSS

To understand whether harness entanglement affects all tasks uniformly or concentrates on particular task types, we compare the base-model episode complexity for tasks that are retained versus lost after cross-harness transfer. For each cross-harness condition, we partition the base-solvable tasks into *retained* tasks (solved by both the base model and the fine-tuned model) and *lost* tasks (solved

by the base model but not by the fine-tuned model under the target harness). We then measure the base-model episode complexity using response length (characters) and tool-call count as proxies.

Table 12 reports the median complexity for retained versus lost tasks across four representative cross-harness cells. In all four cases, lost tasks correspond to base-model episodes that are substantially longer and involve more tool calls. The pattern is consistent across these cells: lost tasks are 19-40% longer and require 25-36% more tool calls than retained tasks.

Table 12: Task complexity comparison: retained vs lost tasks. Complexity measured on base-model episodes. "Retained" = solved by both base and fine-tuned model; "Lost" = solved by base but not by fine-tuned model under target harness.

Condition	**Retained len / calls**	**Lost len / calls**	**Increase**	n **(ret/lost)**
SFT-CC × OPENHANDS	47.0K / 122	64.5K / 164	+37% / +34%	144 / 81
SFT-CC × DEEPSEEK HARNESS	41.1K / 128	57.8K / 168	+40% / +31%	160 / 88
Anchor-preserve × OPENHANDS	45.4K / 106	57.7K / 144	+27% / +36%	92 / 133
Content-preserving × DEEPSEEK HARNESS	44.6K / 134	53.0K / 168	+19% / +25%	194 / 54

In these four conditions, lost tasks have longer base-model episodes and more tool calls than retained tasks. Longer interactions coincide with greater vulnerability, but the table does not identify interaction length as the cause.

D.3 REJECTION ONSET DISTRIBUTION

Rejection onset is pair-dependent. Table 13 reports the first-rejection position in the same diagnostic episodes as the corresponding conditions in Table 9. Each condition contains 495 episodes after retaining the first record for each task. The onset parser splits traces at `<tool_call>`, `<function=`, and `"tool_calls"` markers and associates intervening rejection text with the preceding marker. Indices start at zero, and nested wrappers are counted separately; an index therefore counts parsed markers rather than distinct tool executions.

Among episodes with a located rejection, 65.5% for SFT-DSH×CLAUDE CODE and 56.7% for SFT-OH (transcribed)×CLAUDE CODE first reject by marker index 2. For SFT-CC×DEEPSEEK HARNESS, this share is 4.1% and the median index is 17. The first two conditions reject early in the serialized interaction, whereas SFT-CC on DEEPSEEK HARNESS tends to reject later. These onset summaries use the episodes with a located rejection in this parser's output as their denominator.

Table 13: First-rejection onset from parsed call markers. n is the number of episodes with a located rejection out of 495 per condition. Quartiles and threshold percentages are computed within those n episodes; quartiles use linear interpolation. Indices are zero-based and count nested wrappers separately.

Condition	n	**p25**	**p50**	**p75**	**Index** ≤ 2
SFT-DSH × CLAUDE CODE	113	1	1	3	65.5%
SFT-OH (transcribed) × CLAUDE CODE	263	1	1	3.5	56.7%
SFT-CC × DEEPSEEK HARNESS	367	7	17	36	4.1%

D.4 DETECTOR VALIDITY FOR THE CONDITIONAL ANALYSIS

For the conditional analysis, invalid episodes are identified from explicit runtime rejection messages. Home cells have near-zero rejection rates; DEEPSEEK HARNESS uses a different rejection string from CLAUDE CODE and OPENHANDS, so Table 9 separates the two groups.

Table 14 compares rejected-call rates between lost and retained task groups. Each condition–harness cell uses one retained stochastic episode per task under the unified protocol. The cleanest separation is SFT-DSH on CLAUDE CODE: 41.8% of lost tasks exhibit rejected tool calls versus 12.5% of retained tasks, and SFT-CC-renamed shows the same pattern. Two conditions invert the sign: on CLAUDE CODE, both SFT-OH (transcribed) and SFT-OH (native) show higher rejection rates in

the retained group. One possible explanation is exposure time: retained episodes are longer and accumulate more opportunities for an occasional stray rejected call, while some failed episodes stop before emitting any. Task success and interaction failures can influence one another within an episode, so these conditional rates show association.

Table 14: Association between harness-incompatible behavior and capability loss. "Gap" is the difference between the incompatible-behavior rate in the loss group and the retained-capability group.

Condition	Loss n	Invalid in lost	Invalid in retained	Gap
SFT-DSH × CLAUDE CODE	67	41.8%	12.5%	+29.3 pt
SFT-CC-renamed × CLAUDE CODE	88	29.5%	11.7%	+17.8 pt
SFT-OH (transcribed) × CLAUDE CODE	96	45.8%	69.7%	-23.9 pt
SFT-OH (native) × CLAUDE CODE	52	19.2%	73.4%	-54.2 pt

E MODEL-DEPENDENT SKIN SENSITIVITY

Table 15 tabulates the skin-intervention results of Section 4.3.

Table 15: Interface-skin intervention. Bare DSH, CC skin, and OH skin use the same fixed DEEPSEEK HARNESS runtime and differ only in outer interface representation. Home-reference scores are single-run diagnostics under the native runtime; each skin cell uses one episode per task. The SFT-Mix home reference uses the same diagnostic episodes as its OPENHANDS cell in Table 11.

Model	Bare DSH	CC skin	OH skin	Home reference
Base	50.1	48.1	48.5	–
SFT-CC	41.2	50.7	45.3	53.1 (CC)
SFT-OH (transcribed)	18.0	36.8	49.3	41.0 (OH)
SFT-Mix	44.8	48.9	–	67.9 (OH)
Content-preserving	56.0	52.3	–	52.5 (CC)

For the base model, both skins are below bare DEEPSEEK HARNESS (48.1% / 48.5% versus 50.1%). For SFT-OH (transcribed), the OPENHANDS skin reaches 49.3% while the CLAUDE CODE skin reaches 36.8%, a 12.5-point spread. For SFT-CC, the corresponding spread is 5.4 points (50.7% under the CLAUDE CODE skin versus 45.3% under the OPENHANDS skin). For the content-preserving paired construction, the reported comparison is between the CLAUDE CODE skin (52.3%) and bare DEEPSEEK HARNESS (56.0%).

The 12.5- and 5.4-point spreads show that sensitivity to skin identity differs between the two single-harness models. Interface conventions, source trajectory behavior, and unmatched transcription events can all contribute.

F BASE-MODEL PRIOR PROBE

The paired cross-entropy probe of Section 4.5 compares matched trajectory content rendered in the CLAUDE CODE and OPENHANDS dialects. Reasoning content is backfilled where necessary, and final responses are paired to reduce content differences while retaining harness-specific rendering. Across 40 pairs, the mean paired difference $\mathrm{CE}_{\text{CLAUDE CODE}} - \mathrm{CE}_{\text{OPENHANDS}}$ is -0.0265 nats. The share of pairs for which CLAUDE CODE is harder is 0.500. The paired summaries are near zero and balanced; 40 pairs cannot resolve weak differences.

G PAIRED-MASK COVERAGE AUDIT

Content-preserving data and packing. The content-preserving construction begins with 2,682 distinct source–renamed pairs (5,364 rows). Atomic packing at the 262,144-token limit retains 1,827 pairs and excludes the 855 pairs that exceed the joint limit. Both renderings of each retained pair receive SFT. Of the retained pairs, 165 have unequal tokenized lengths and are SFT-only; the

remaining 1,662 equal-length pairs are eligible for consistency. These are counts of distinct pairs. The training packs contain two passes over the retained data: 3,654 pair presentations (7,308 rows), comprising 3,324 equal-length and 330 SFT-only pair presentations.

Candidate renamed identifiers are sampled to match the original identifier's tokenized length in probe contexts. The renaming procedure replaces a tool or parameter identifier only when a length-preserving tokenization exists; 23–29% of tool names per packed trajectory are actually renamed. For each consistency-eligible pair, M contains jointly supervised target positions whose target-token IDs agree at corresponding positions within each row. Across the 1,662 eligible pairs, M contains 8,703,313 of 49,674,405 jointly supervised target positions (17.5%, token-weighted). Of these, 277 of the 1,662 eligible pairs (16.7%) have at least 99% matched-supervision coverage. This is a per-pair mask statistic, not an end-to-end token-alignment claim. BPE boundary shifts can make equal target-token IDs at corresponding row positions coincidental.

Content-preserving and its $\lambda \in \{0, 0.5, 2\}$ controls share the same original paired corpus, retained pairs, and packing schedule; only the consistency weight differs. At λ=0, the no-consistency control supervises all retained rows through SFT alone. The shuffled variants keep the cross-task packing and improve in sequence as consistency is added; their three-evaluation means are shown in Figure 4, and Table 16 compares the paired and shuffled ablations under the same reporting convention.

The teacher-forced fresh-rename NLL gap is nearly unchanged between the content-preserving no-consistency and consistency arms (0.0056 versus 0.0057), despite their different decoded task-success scores.

Shuffled masks and loss aggregation. The three shuffled arms use identical packed data and SFT targets, including the same pairing permutation and row order. Row boundaries are identified by position-index resets, and adjacent rows form a pair. For unequal-length pairs, only the consistency comparison is restricted to the shorter row's length; all supervised tokens in both rows contribute to SFT. ARCHER's target-ID-matched mask selects jointly supervised positions with equal target-token IDs. The random-mask control samples uniformly without replacement from all jointly supervised positions in that window, using exactly the same number of positions for each pair. Its random seed is determined by the pair's row offsets, so a given packed pair uses the same mask across epochs. This matches cardinality, while token types and locations can differ. Pairs with an empty mask contribute only to SFT.

Equation 4 applies a common supervised-token denominator to the SFT and consistency sums within each microbatch; the training loss averages these values over microbatches. The consistency sum has no additional division by the number of selected sites. Gradients flow through both token losses without detachment or loss clipping; the optimizer uses the global gradient clipping specified in Appendix A.

G.1 Position-Level Ablations

Table 16 reports the content-preserving and shuffled ablations on held-out DeepSeek Harness. It compares the two data constructions and the consistency variants within the shuffled construction. The content-preserving $\lambda \in \{0.5, 1, 2\}$ sweep is a retrospective sensitivity analysis of the default λ=1, evaluated on the same 495-task pool as the main results. It uses no separate development split and did not set ARCHER's λ. All rows below summarize three independent evaluations.

Shuffled-pair attribution. In the repeated-evaluation results (Tables 1 and 2), Shuffled-NC exceeds the content-preserving arm on both held-out harnesses, and both shuffled consistency variants improve further. The first comparison concerns the full data construction, including packing and retained context; the latter comparisons concern consistency within that construction. ARCHER's advantage over random positions is a difference in observed means, not evidence of semantic alignment at the selected sites.

G.2 High-Coverage Counterfactual

A separately regenerated, same-trajectory paired corpus increases token-weighted matched-supervision coverage from 17.5% to 88%. Its renaming procedure searches for identifier replace-

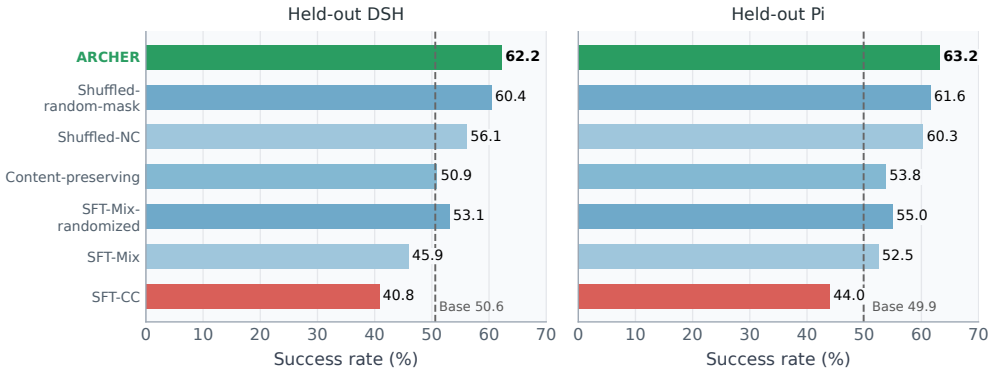


Figure 4: Held-out performance across training constructions. The three shuffled bars compare no consistency, random-position consistency, and ARCHER; the other bars show reference SFT constructions. Dashed lines mark the corresponding base-model scores. Bar heights are means over three independent evaluations; Tables 1 and 2 report the corresponding means and sample SDs.

Table 16: Pairing, consistency-weight, and mask ablations on held-out DEEPSEEK HARNESS (success rate, %). The four content-preserving rows share the same paired data and packing, varying only λ; the shuffled consistency arms use λ=1, and NC uses λ=0. Entries are mean $\pm$ sample SD over three independent evaluations.

Condition	**DEEPSEEK HARNESS**
Content-preserving-NC	49.1±3.0
Content-preserving (λ=2)	49.8±1.6
Content-preserving (λ=0.5)	53.2±1.8
Content-preserving (λ=1)	50.9±2.2
Shuffled-NC	56.1±0.9
Shuffled-random-mask	60.4±1.2
ARCHER	**62.2±2.7**

ments that preserve token positions; where the intended alignment cannot be obtained, it falls back to the original content-preserving renamed view. The resulting raw corpus contains all 2,682 source pairs before packing. This corpus is used only for the high-coverage counterfactual; the primary content-preserving result and its λ controls use the original corpus audited above.

Table 17 reports the coverage and results. The construction changes the paired examples, mask cardinality, and total consistency pressure together, so the table compares constructions rather than isolating coverage or normalization. The $|M|$-normalized condition is an additional normalization ablation of this high-coverage content-preserving construction. The unequal-target mask instead selects jointly supervised positions with $y_t^A \neq y_t^B$, complementing the default equal-target rule. It uses the same data, packing, and λ=1 as the default high-coverage arm. Of the 107,924,804 jointly supervised positions, 95,008,030 (88.0%) have equal target IDs and 12,916,774 (12.0%) have unequal target IDs.

G.3 SINGLE-HARNESS PAIRS

The single-harness paired variants fail to transfer to held-out DEEPSEEK HARNESS: CLAUDE CODE-only reaches 41.2% and OPENHANDS-only reaches 33.7%, both below the unpaired two-harness mixture (44.8%) and below ARCHER (64.0%). The corresponding full matrices are 44.8% / 29.5% / 41.2% for CLAUDE CODE-only and 46.1% / 55.6% / 33.7% for OPENHANDS-only on CLAUDE CODE/ OPENHANDS/ DEEPSEEK HARNESS. All values in this appendix comparison are single-run diagnostics. The tested single-harness paired variants are insufficient in this setup.

Table 17: High-coverage counterfactual using a separately regenerated paired corpus (success rate, %). The original content-preserving reference reports mean ± sample SD over three independent evaluations; high-coverage rows are single-run point estimates. The coverage column reports the fraction of jointly supervised positions with equal target IDs in the underlying corpus; the unequal-target mask selects the complementary 12.0%.

Variant	Matched-target coverage	DEEPSEEK HARNESS
Content-preserving	17.5%	50.9±2.2
High-coverage pair (λ=1.0)	88%	46.5
High-coverage pair (λ=0.2)	88%	50.3
High-coverage pair (λ=0.5)	88%	47.9
High-coverage pair, no consistency	88%	40.4
High-coverage pair, normalized by $\lvert M\rvert$	88%	50.1
High-coverage pair, unequal-target mask	88%	50.1

G.4 CONSISTENCY-MASKED POSITION ANALYSIS

In the content-preserving audit, the 299,847 matched, jointly supervised positions from pairs below the 99% per-pair coverage threshold decompose as: 40% numeric literals, 16% whitespace and newlines, 17% punctuation, and the remainder other content tokens. The top-5 most frequently matched tokens are `'0'`, `'\n'`, `','`, `'1'`, and `'2'`. Within this lower-coverage subset, equal IDs at corresponding row positions can be coincidental after BPE boundary shifts; the matched tokens are mostly numeric literals, whitespace, and punctuation. This analysis describes that content-preserving subset, not the realized mask of shuffled ARCHER. Within the subset, matching token IDs is not evidence of identifier, semantic, or action alignment.

H THREE-HARNESS MIXTURE REFERENCE

A reference arm mixes natively collected CLAUDE CODE +OPENHANDS +DEEPSEEK HARNESS trajectories (*three-harness mixture*). In a single diagnostic evaluation it scores 48.9% / 50.9% / 63.2% on CLAUDE CODE/ OPENHANDS/ DEEPSEEK HARNESS. Its DEEPSEEK HARNESS score is a *same-side* cell – DEEPSEEK HARNESS appears in its training mixture – so it is a same-side reference. Relative to the single-run two-harness mixture, the three-harness reference raises the DEEPSEEK HARNESS point estimate from 44.8% to 63.2%, lowers OPENHANDS from 67.9% to 50.9%, and leaves CLAUDE CODE similar (48.7% vs. 48.9%). The three-harness arm exposes the coverage trade-off in this mixture; in the same single diagnostic evaluation, ARCHER reaches 64.0% on DEEPSEEK HARNESS without using DEEPSEEK HARNESS training data and reaches 53.4% / 68.9% on CLAUDE CODE/ OPENHANDS.

I HARNESS INTERFACE SIZE

Table 18 gives the character and token counts behind Section 4.3. The corresponding numbers of tools are 23 for CLAUDE CODE, 6 for OPENHANDS, and 12 for DEEPSEEK HARNESS. The interfaces differ substantially in prompt and schema size; with three harnesses, these counts do not explain transfer.

Table 18: Interface components of the three training harnesses.

Harness	System chars	Schema chars	System tokens	Schema tokens
CLAUDE CODE	7.3K	70.8K	1,658	17.8K
OPENHANDS	12.3K	20.9K	2,703	4.7K
DEEPSEEK HARNESS	1.9K	9.3K	386	2.2K

J LIMITATIONS

The study covers four harnesses, one primary base model, and one teacher. We also attempted Gemma 4 E4B and Qwen3.5-2B, but their zero and single-digit success rates, respectively, did not permit informative cross-harness comparisons; they are therefore exploratory negative results rather than evidence for or against ARCHER.

Training and evaluation task identifiers are disjoint, but repositories overlap, so the transfer result is not a test on unseen repositories. Pairing changes which examples survive atomic packing; shuffled pairs additionally use shared-prefix consistency windows while retaining full-row SFT supervision. The ordinary mixture uses per-sequence packing with matched token budgets and schedules. Some diagnostics are interface-specific, and the conditional failure analysis is associational rather than causal.

We focus on SFT because it provides a controlled and resource-efficient setting for isolating harness-dependent behavior under a fixed training recipe. Whether RL training, with its online interaction and reward-driven updates, produces a different form or degree of harness entanglement remains an important direction for future work.