# MLTOOLBENCH: LEARNING TOOL-AUGMENTED AGENTS FOR MACHINE LEARNING DEVELOPMENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Machine learning engineering (MLE) agents have made substantial progress, but learning through ML experimentation remains costly in time and computation. Synthetic environments reduce these costs while introducing variations in data and experimental settings that require task-specific diagnosis. Access to diagnostic tools alone does not ensure that agents learn when to use them or how to act on their findings. We introduce MLToolBench, a suite of executable tools for data inspection, code verification, and experiment diagnosis, together with an SFT and RL pipeline for learning their use. Diagnostic calls acquire evidence whose value depends on subsequent decisions, so final outcomes provide limited guidance on which calls to reinforce. We address this challenge with SPICE, which measures how privileged context changes the likelihood of a sampled tool action and uses this difference as a turn-level reward alongside the final outcome. We train on 80 synthetic tasks and evaluate on 25 in-domain and 10 out-of-domain tasks. Providing tool interfaces and descriptions alone yields inconsistent gains across unadapted models. With the same diagnostic interface, our training pipeline raises in-domain success from 24.8% to 52.4% for Qwen3-8B and from 35.6% to 69.2% for Qwen3.5-35B-A3B. The latter also improves from 31% to 48% out-of-domain, supporting learned diagnostic tool use on held-out sources and targets.

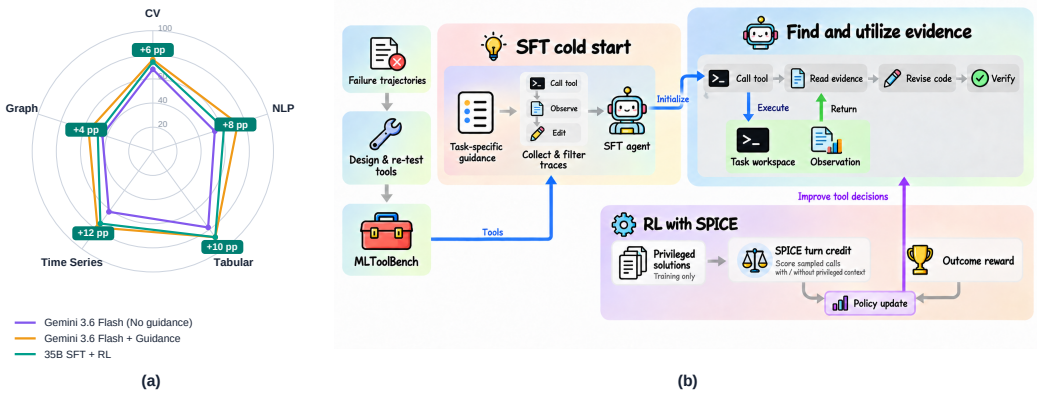



Figure 1: **Learning to find and utilize diagnostic evidence.** **(a)** In-domain success under three setups: Gemini 3.6 Flash with and without guidance, and Qwen3.5-35B-A3B after SFT + RL, all using MLToolBench. Guided Gemini receives solution-informed hints. Signed pp labels show Qwen's gains over unguided Gemini. **(b)** Failure-driven tool construction, guided SFT, and RL with SPICE teach agents to acquire and use diagnostic evidence.

## 1 INTRODUCTION

Machine learning engineering (MLE) agents increasingly perform multi-step ML development tasks (Liu et al., 2026b; Yang et al., 2026b). Learning from these interactions is costly. Verifying a



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.

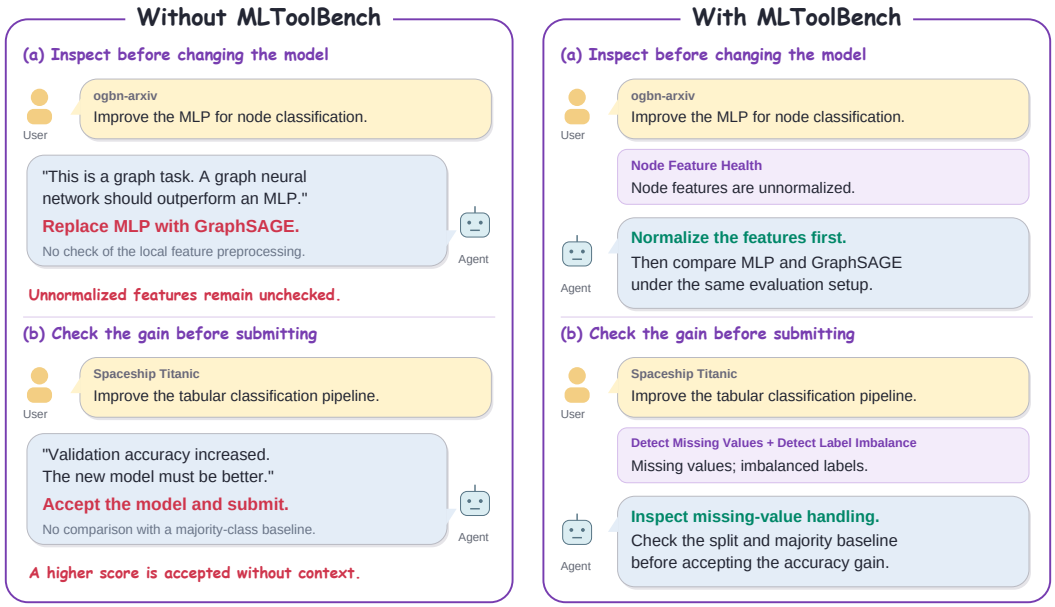



Figure 2: Schematic motivating cases for MLToolBench. **Top:** local feature diagnostics guide preprocessing checks before an architectural change. **Bottom:** missingness and imbalance motivate checking an apparent validation gain before submission. The contrast illustrates why evidence acquisition should precede intervention; Section 3 separately evaluates aggregate task performance.

single rollout may require preprocessing data, training a model, and evaluating its output. Repeating this process places substantial demands on time and computation (Cai et al., 2026a). Recent work develops verifiable tasks and synthetic environments for MLE agents, with small synthetic datasets reducing the cost of execution and verification during RL rollouts (Qiang et al., 2026b; Zhou et al., 2026). Synthetic task generation can also be combined with pass-rate-based task selection to support RL curricula that provide challenges suited to the model's current capabilities (The Microsoft AI Team, 2026).

These environments make interaction more affordable, but agents must still diagnose the task they actually encounter. A synthetic instance may resemble a familiar ML problem while changing its data distribution, preprocessing, or evaluation conditions. Two failure patterns illustrate this challenge. First, **mismatched modeling priors** arise when agents select code modifications based on familiar strategies without inspecting the local data. In Figure 2 (top), the agent replaces an MLP with GraphSAGE before checking whether node features are normalized. Second, **insufficient baseline comparison** occurs when agents treat a validation score as evidence of improvement without comparing it against a simple reference model. In Figure 2 (bottom), class imbalance makes comparison with a majority-class baseline essential before accepting the apparent gain. These concerns extend beyond synthetic environments. Real-world MLE tasks span diverse datasets and experimental requirements, making it difficult to anticipate every check an agent will need. MLAgentBench reports agents discarding features before assessing their predictive value (Huang et al., 2024), while MLE-bench documents failures to use available submission checks, resulting in invalid outputs (Chan et al., 2025). Across these settings, the common challenge is to connect task-specific evidence to the next decision. Execution tools make checks possible, but agents must learn which checks to perform and how to use their results. We therefore ask: *How can MLE agents learn to find and utilize diagnostic evidence effectively?*

In this work, we introduce MLToolBench and a training framework that jointly address this capability gap. Built on the MLAgentBench scaffold, MLToolBench exposes data inspection, code verification, and experiment diagnosis as explicit actions, with tools derived from recurring agent failures. Agents acquire evidence by invoking these tools; their interfaces and descriptions alone provide no

practical experience in interpreting the results. Guided SFT establishes inspection-to-action behavior, and RL refines it through task interaction. The tool library and learning method thus serve the same objective: learning to obtain diagnostic evidence and use it to guide ML development.

Diagnostic tool use creates a particular credit-assignment challenge. Its benefit often emerges through a later decision, such as normalizing features after inspecting their scale. Final task rewards do not identify which earlier inspections informed that decision. Potential-based methods such as TIPS track changes in reference-answer likelihood (Xie et al., 2026), yet useful diagnostic evidence need not immediately increase that likelihood. On fixed SFT trajectories, we observe smaller mean absolute TIPS scores for diagnostic calls than for editing/execution actions across three scoring models (Figure 4). This finding concerns feedback magnitude, not causal credit. Motivated by this distinction, we propose *Solution-Privileged Information Credit Estimation* (SPICE), inspired by knowledge distillation with privileged information (Lopez-Paz et al., 2016). A shared frozen model scores each sampled action with and without a verified solution summary. The resulting log-likelihood difference measures contextual support for the action and supplies signed turn-level feedback alongside the outcome reward. This gives diagnostic decisions a learning signal without requiring an immediate increase in reference-answer likelihood.

Our contributions are as follows:

- **An executable diagnostic interface for MLE.** We construct MLToolBench from recurring agent failures, exposing data, code, and experiment checks as explicit actions within MLAgentBench. Task reruns verify candidate tools before their inclusion in the library.
- **A training framework for diagnostic decisions.** Guided SFT establishes initial tool-use behavior. RL then combines task outcomes with SPICE, which scores sampled actions using the change in their likelihood under privileged solution context.
- **Evidence for learned diagnostic tool use.** Across held-out ID and OOD tasks, the full pipeline improves both Qwen backbones. Interface comparisons, matched RL controls, and diagnostic-observation interventions examine tool access, training, and evidence use separately.

## 2 METHOD

**Method overview.** Our pipeline connects tool construction, demonstration learning, and reinforcement learning (Figure 1b). MLToolBench makes diagnostic evidence accessible through executable calls. Guided SFT teaches an initial pattern of inspection and intervention; RL refines the policy using task outcomes and SPICE turn-level feedback. The trained student acquires observations from the workspace and receives no privileged solution context at evaluation.

**Notation.** A task $x$ contains an initial workspace, instruction, and executable evaluator; $\mathcal{T}_x$ denotes its workspace and diagnostic actions. At interaction index $s$, policy $\pi_\theta$ generates an assistant action $a_s$ from history $h_{s-1}$ and receives an observation $o_s$. We call $z_s = (a_s, o_s)$ a segment and write $h_s = (x, \mathcal{T}_x, z_{1:s})$ for the resulting history. A rollout is $\tau = (z_1, \ldots, z_S)$. We use $t$ for a token position within assistant-generated text, with token $y_t$ and ordinary prefix $H_t$. The evaluator assigns an outcome $R(\tau)$ to the rollout; SPICE scores sampled tool calls to construct turn-level rewards. SFT and RL optimize the same parameters $\theta$.

### 2.1 MLTOOLBENCH: DIAGNOSTIC INTERFACE CONSTRUCTION

Our benchmark pairs a diagnostic tool library with executable ML task instances for training and evaluation. MLToolBench extends MLAgentBench's workspace interface with 67 registered diagnostic tools: seven shared tools and 60 tools organized into tabular, time-series, NLP, vision, and graph modules (Huang et al., 2024). They support three functions: *data inspection* (e.g., missing values, label balance, and node-feature health), *code verification* (e.g., configuration consistency, tensor shapes, and output formats), and *experiment diagnosis* (e.g., metric extraction, trends, and plateaus). Time-series tools also analyze training curves across domains. Figure 3 summarizes construction: we identify missing evidence in failed trajectories, design a diagnostic, and re-run the original task with the candidate tool. Candidates are retained when the original failure is resolved; otherwise, they are refined and re-tested. Agents call these tools to obtain measurements and, in

some cases, heuristic suggestions, then choose subsequent edits themselves. Appendix C lists the complete library and distinguishes it from each run's exposed tools.

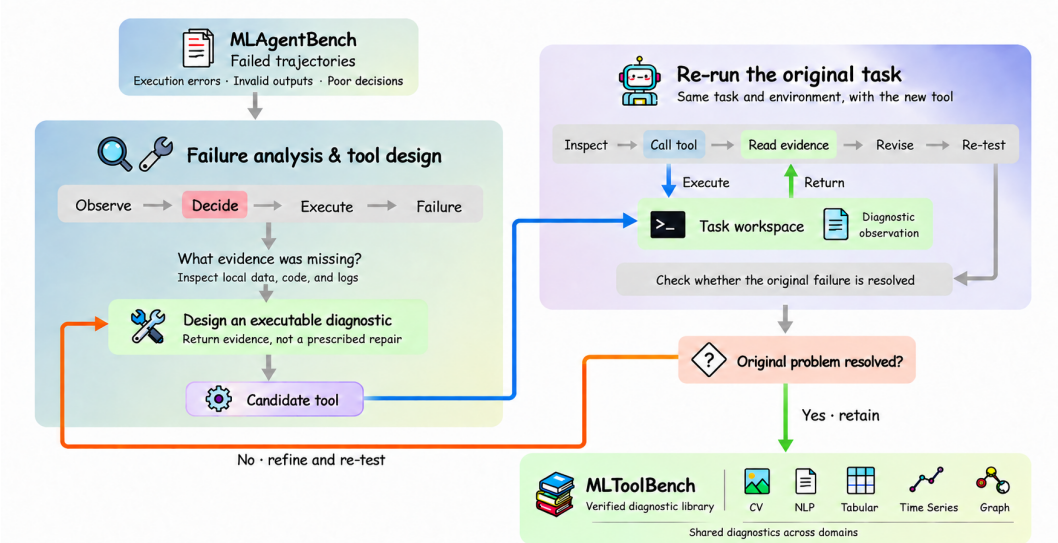



Figure 3: **Construction of MLToolBench.** Failure analysis motivates diagnostic tool candidates. The agent re-runs the original task with each candidate; tools that help resolve the original failure are retained and organized by domain, while unsuccessful candidates are refined and re-tested.

### 2.2 Tool-Use Cold Start via Supervised Fine-Tuning

**Constructing task-specific guidance.** We use an LLM to assess each training task from its specification, reference solution, and synthetic environment generation mechanism. It tailors guidance to the instance's prediction target, data characteristics, starter pipeline, and evaluation constraints, specifying which diagnostic tools to call and how alternative observations should inform subsequent actions. The guidance provides an investigation strategy rather than a target patch; the agent must still invoke tools to obtain evidence from the workspace.

**Collecting demonstrations for a tool-use cold start.** We insert this guidance into Gemini 3.6 Flash's prompt alongside the task and available tool descriptions. The model then interacts with the environment to produce demonstrations that connect tool calls and observed evidence to subsequent actions. We filter these trajectories and use them for SFT, removing the guidance from student inputs while retaining the actual interactions. Standard next-token supervision is applied only to assistant-generated tokens; tool observations remain context. This gives the policy an initial ability to use diagnostic tools before RL. Appendix D.1 provides collection, filtering, and training details.

### 2.3 Reinforcement Learning with Turn-Level Credit

After SFT, the policy learns from fresh interactions without collection-time guidance. Final task rewards evaluate the resulting ML artifact, but provide limited feedback on diagnostic decisions along the way. Checking feature scales, for example, acquires evidence whose value depends on a later preprocessing choice. We therefore propose *Solution-Privileged Information Credit Estimation* (SPICE). Inspired by knowledge distillation with privileged information (Lopez-Paz et al., 2016), SPICE uses solution summaries as additional scoring context. The policy acts without these summaries. Their effect on the likelihood of its sampled action supplies a turn-level reward.

**From solution progress to action support.** TIPS scores changes in reference-answer likelihood across an interaction (Xie et al., 2026); our adaptation uses solution summaries as reference answers. Diagnostic evidence can guide later ML decisions without immediately increasing their likelihood. SPICE instead asks whether a verified solution summary increases support for the sampled tool

action in its current history. Comparing the same action with and without that context subtracts its ordinary log-likelihood and supplies a complementary credit proxy, without requiring immediate progress toward predicting the reference answer.

**Scoring tool-use turns.** Each tool-use turn contains an assistant tool call and the environment's returned observation. For a sampled call $a_{i,s}$ in rollout $i$, turn $s$, we compare its likelihood under the same frozen policy snapshot $q$, with and without verified-solution context. Let $h_{i,s-1}$ be the pre-action history and $\mathcal{C}_x$ a frozen set of $K$ agent-written summaries of verified candidate solutions for the current training task. The turn score is

$$d_{i,s} = \frac{1}{K|a_{i,s}|} \sum_{c \in \mathcal{C}_x} \sum_{u=1}^{|a_{i,s}|} \log \frac{q(a_{i,s,u} \mid h_{i,s-1}, c, a_{i,s,<u})}{q(a_{i,s,u} \mid h_{i,s-1}, a_{i,s,<u})}. \tag{1}$$

Both branches teacher-force the same sampled tool-call tokens. The current tool observation and future trajectory are excluded from scoring; earlier observations remain in the history. Positive and negative scores indicate increased and decreased support under privileged context. The solution bank starts from verified demonstrations and admits newly verified RL solutions between updates. Each call is scored in its own history, so trajectories need not share tools or turn counts.

**Adding turn rewards to outcome feedback.** We use binary terminal success $R_i$ under Equation 5, extending MLAgentBench's threshold (Huang et al., 2024) to signed and near-zero baselines with an absolute-gain floor of 0.01. The task test script evaluates submissions and starter baselines on the same partition; invalid submissions receive zero. We add the detached tool score at its turn boundary and $R_i$ at termination. With $S_i$ turns and shaping coefficient $\beta \geq 0$, the reward and undiscounted return-to-go are

$$r_{i,s} = \mathbf{1}[s = S_i] R_i + \beta \, \mathrm{sg}(d_{i,s}), \qquad G_{i,s} = \sum_{j=s}^{S_i} r_{i,j}. \tag{2}$$

Non-tool turns and calls without a valid reference score receive zero shaping reward. To control shaping magnitude, we calibrate $\beta$ with a short training-task pilot for each backbone to target a mean absolute turn-level shaping reward in $[0.05, 0.3]$ on pilot rollouts (Appendix D.2).

**Advantage estimation and policy update.** We adapt GRPO (Shao et al., 2024) to turn-level returns. For $B$ rollouts of task $x$, we center each turn's return using the mean complete return:

$$\mu_x = \frac{1}{B} \sum_{i=1}^{B} G_{i,1}, \qquad \widehat{A}_{i,s} = G_{i,s} - \mu_x. \tag{3}$$

We omit group standard-deviation scaling to avoid the difficulty-dependent task weighting identified by Liu et al. (2025). Let $\mathcal{P}_i$ contain assistant tokens, $s(i,t)$ identify token $y_{i,t}$'s turn, and $\rho_{i,t} = \pi_\theta(y_{i,t} \mid H_{i,t}) / \pi_{\mathrm{old}}(y_{i,t} \mid H_{i,t})$, where $H_{i,t}$ is its original policy context. We minimize the clipped objective

$$\mathcal{L} = -\mathbb{E}\left[ \frac{1}{B} \sum_{i=1}^{B} \frac{1}{|\mathcal{P}_i|} \sum_{t \in \mathcal{P}_i} \min\left\{ \rho_{i,t} \widehat{A}_{i,s(i,t)}, \mathrm{clip}(\rho_{i,t}, 1-\eta, 1+\eta) \widehat{A}_{i,s(i,t)} \right\} \right]. \tag{4}$$

Here $\eta$ is the clipping threshold. Scores and advantages are detached; tool observations are context, not prediction targets. Setting $\beta = 0$ recovers outcome-only training with mean-centered advantages. Appendix D.2 specifies token assignment and reference scoring.

## 3 Experiments

We ask four questions: (Q1) Does learning convert diagnostic access into task success? (Q2) Do gains transfer to held-out sources and targets, and does diagnostic evidence affect success? (Q3) How do training and credit assignment change tool-use behavior and feedback? (Q4) What do guided demonstrations, RL, and SPICE each contribute?

### 3.1 EXPERIMENTAL SETUP

**Benchmark and task splits.** We use 80 training, 25 ID, and 10 OOD instances across vision, NLP, tabular learning, time series, and graph learning. ID reuses training templates with held-out data or generated samples; OOD holds out data sources or prediction targets within these domains. Multiple instances can derive from one source (Qiang et al., 2026b; Zhou et al., 2026). Evaluation instances are excluded from SFT, solution banks, and RL. Appendix F defines all instances and splits.

**Evaluation protocol.** We report success rates using a signed-baseline extension of MLAgentBench's improvement threshold (Huang et al., 2024): valid submissions must achieve a direction-adjusted improvement exceeding both 10% of the baseline magnitude and an absolute-gain floor of 0.01 (Appendix A.1). Failed or invalid runs are unsuccessful. For each fixed model or checkpoint, we run ten evaluation rollouts per task, yielding 250 ID and 100 OOD runs per condition; no retraining occurs between these repeats. *Original* denotes the MLAgentBench scaffold with its standard tools for file editing and code execution, without MLToolBench diagnostics; + *MLToolBench* adds these diagnostics. These labels specify evaluation-time tool access, irrespective of training condition. Tasks and execution budgets are held fixed.

**Compared methods.** Table 1 compares closed-source Gemini 3.6 Flash and Claude Sonnet 5 (Anthropic, 2026) with open-source coding and Qwen baselines, including DeepSeek-Coder-33B-Instruct (Guo et al., 2024), under the same scaffold. Unadapted baselines receive tool interfaces and descriptions, but no demonstrations, task-specific guidance, or prior training on this interface; they can call tools and observe outputs during evaluation. Both Qwen backbones receive full SFT and RL with SPICE; full-SFT controls without MLToolBench demonstrations appear in the ablation study (Appendix Figure 6). Learned policies receive no collection-time guidance at evaluation.

**Training and objective controls.** Each SFT condition collects 100 Gemini trajectories per training task (8,000 before filtering). RL continues tool-enabled full SFT for 100 updates with fresh on-policy rollouts; we evaluate the final checkpoint. Equation 2 combines outcome and signed turn rewards. For each backbone, outcome-only and TIPS controls share the SFT initialization, training data, tool interface, rollout budget, optimization settings, and binary outcome reward with SPICE. Outcome-only sets $\beta = 0$; TIPS replaces the shaping method. Appendix A describes the evaluation protocol and training configuration; Appendices D.1 and E detail SFT and prompts.

### 3.2 MAIN RESULTS

**Training converts diagnostic access into task success.** With MLToolBench, SFT followed by RL raises ID success from 24.8% to 52.4% for Qwen3-8B and from 35.6% to 69.2% for Qwen3.5-35B-A3B (Table 1, Q1). The 35B policy exceeds augmented-interface Claude and unguided Gemini by 6.8 and 8 points, but remains below privileged guided Gemini (74.4%). These 27.6/33.6-point gains demonstrate policy improvement under an unchanged diagnostic interface.

**Tool descriptions alone do not ensure effective use.** Without tool-specific training or guidance, diagnostic access reduces ID success for Gemini, Claude, and both coding baselines; unadapted Qwen3-8B and Qwen3.5-35B-A3B gain only 0.4/1.2 points. For the SFT + RL policies, retaining diagnostics at evaluation raises ID success by 3.2/18 points for 8B/35B. Original measures interface removal after tool-enabled training; Section 3.5 separately evaluates SFT without MLToolBench. This contrast motivates learning to use diagnostics beyond exposing their interfaces.

### 3.3 GENERALIZATION AND DIAGNOSTIC UTILITY

**Generalization.** On held-out sources and prediction targets, SFT followed by RL raises OOD success with MLToolBench from 20% to 36% for 8B and from 31% to 48% for 35B (Table 1). Diagnostic access improves the trained policies by 8 and 7 points, respectively. The 35B policy attains 48%, compared with Claude at 43% and Gemini at 46% under the same augmented interface. This supports within-suite transfer, but 100 OOD runs leave broader generalization uncertain.

**Diagnostic utility.** Table 6 compares diagnostic-observation interventions across all 250 ID rollouts per policy. Masking lowers success by 1.6 points for outcome-only and 14.8 for SPICE, whereas paraphrasing lowers it by 0.4 and 0.8 points. For SPICE, masking removes 37 successes versus two under paraphrasing. Both retain tool access, isolating an observation intervention rather than

| Backbone | Model / training | In-domain | | Out-of-domain | |
|---|---|---|---|---|---|
| | | Original | + MLToolBench | Original | + MLToolBench |
| *Closed-Source Models* | | | | | |
| Gemini 3.6 Flash | | **65.20%** | 61.20% | **52.00%** | 46.00% |
| Claude Sonnet 5 | | 64.80% | 62.40% | 45.00% | 43.00% |
| *Open-Source Models* | | | | | |
| Qwen2.5-Coder-14B | | 45.20% | 40.40% | 34.00% | 32.00% |
| DeepSeek-Coder-33B-Instruct | | 57.20% | 49.20% | 36.00% | 34.00% |
| Qwen3-8B | | 24.40% | 24.80% | 23.00% | 20.00% |
| Qwen3.5-35B-A3B | | 34.40% | 35.60% | 29.00% | 31.00% |
| *Open-Source Reasoning Models (Fine-Tuned)* | | | | | |
| Qwen3-8B | Full SFT | 36.80% | 38.40% | 28.00% | 32.00% |
| | SFT + RL (SPICE) | 49.20% | 52.40% | 28.00% | 36.00% |
| Qwen3.5-35B-A3B | Full SFT | 39.60% | 44.40% | 34.00% | 41.00% |
| | SFT + RL (SPICE) | 51.20% | **69.20%** | 41.00% | **48.00%** |

Table 1: Main success rates (%; 250 ID and 100 OOD runs per condition). Fine-tuned models use MLToolBench during training. Original uses standard MLAgentBench tools without MLToolBench diagnostics; + MLToolBench adds these diagnostics at evaluation. Bold marks column maxima.

interface removal. However, policies visit different states and invoke different tools; these effects do not isolate use of the same evidence or credit accuracy.

### 3.4 Credit Assignment and Tool-Use Behavior

**Training broadens diagnostic tool use.** For Q3, 35B data-management/verification coverage rises from 0.3/0.5 to 2.7/4.7 rollouts out of five (Appendix Figure 8). The 8B RL policy achieves higher success with similar coverage but more distinct tools (six versus two). For 35B, SPICE uses eight distinct MLToolBench tools in 11 calls on average, versus two in nine calls for outcome-only RL (Appendix Table 5). TIPS averages three tools in ten calls. The comparable call counts but different numbers of distinct tools suggest that SPICE broadens diagnostic selection, beyond simply increasing invocation frequency. These counts do not establish whether repeated calls are useful.

**Feedback magnitude across action types.** Figure 4 reports mean absolute segment scores on fixed SFT trajectories, using three frozen scorers and a shared solution bank. The diagnostic-to-editing/execution ratio is 0.37–0.54 for TIPS versus 0.80–0.93 for SPICE, indicating a smaller relative magnitude gap. Absolute values discard sign and establish neither score correctness nor alignment with observation utility. Training outcomes are evaluated separately in Section 3.5.

### 3.5 Ablation Studies

For Q4, we follow the pipeline from guidance and SFT to RL and reward design. Training-time tool dropout is reported in Appendix B.1.

**Guidance and diagnostic access.** We first examine guidance for Gemini (Figure 5). Diagnostics alone reduce ID success from 65.2% to 61.2%, whereas guidance alone raises it to 68.4%. Combining both reaches 74.4%. OOD success follows the same pattern: 52%, 46%, 56%, and 65%, respectively. This motivates teaching inspection strategies alongside tool access. Guided Gemini is a privileged reference receiving solution-informed hints without environment feedback (Appendix E).

**Tool-use supervision.** We next examine SFT. With the augmented evaluation interface held fixed, tool-enabled demonstrations improve ID/OOD success over demonstrations collected without MLToolBench by 13.2/12 points for 8B and 5.2/9 for 35B (Appendix Figure 6). This comparison evaluates the demonstration pipeline, including collection-time guidance, with equal raw trajec-

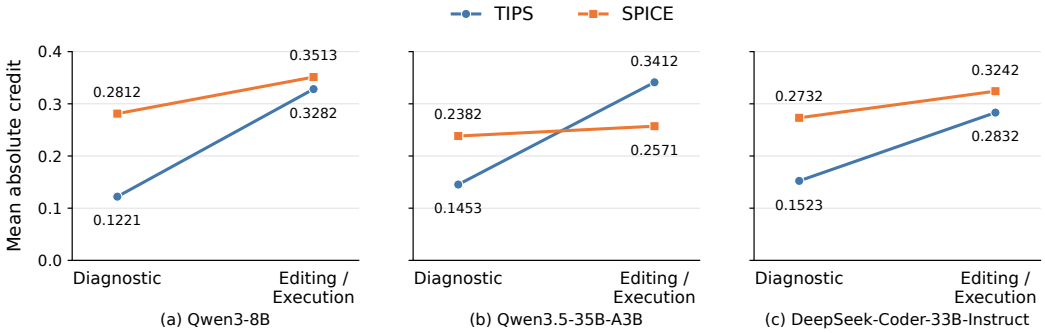
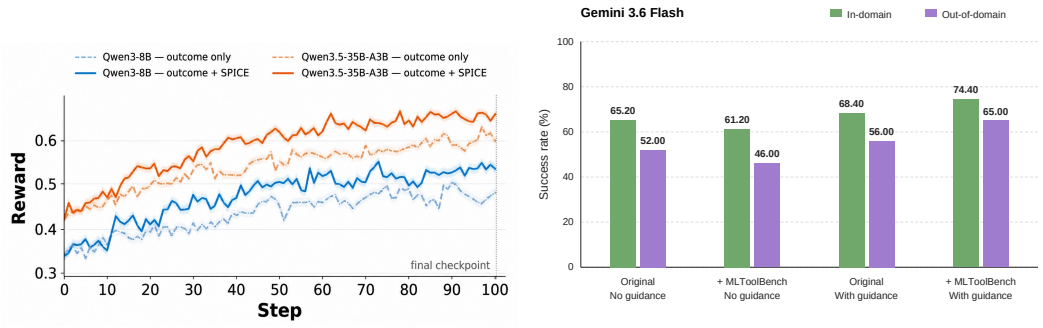



Figure 4: **Feedback magnitude on fixed SFT trajectories.** Mean absolute unscaled scores: TIPS potential differences and SPICE $d_{i,s}$. Panels use frozen, task-unadapted scorers; raw scales differ. Editing/execution includes code edits and execution. Absolute values neither reveal score signs nor establish correct credit assignment.




**(a)** Training reward dynamics **(b)** Guidance and diagnostic access

Figure 5: **Training and guidance ablations.** (a) Mean binary outcome reward $R_i$ for 8B (blue) and 35B (orange): outcome-only GRPO (dashed) and SPICE (solid). Both curves exclude shaping rewards. Final checkpoint: step 100. (b) Gemini ID/OOD success (%); unguided results match Table 1. Training outcome success is distinct from held-out success.

tory budgets before filtering. These gains support the combined supervision pipeline; guidance and demonstration content are not isolated.

**SPICE improves over matched RL objectives.** RL with SPICE improves over full SFT by 14/4 ID/OOD points for 8B and 24.8/7 for 35B. Matched RL comparisons hold initialization, data, tools, and optimization settings fixed. SPICE exceeds outcome-only RL by 13.2/3 points for 8B and 21.2/4 for 35B, and TIPS by 8/1 and 14/3 points, respectively (Appendix Figure 7). TIPS and SPICE share solution summaries, controlling reference content while changing how it supplies turn-level feedback. The 35B ID gaps correspond to 53 additional successes over outcome-only RL and 35 over TIPS among 250 evaluations. Both backbones favor SPICE under matched conditions, although fixed-checkpoint repeats do not establish training-seed robustness or statistical significance.

**Action–feedback correspondence contributes to the gain.** Within-task, within-category score shuffling preserves each permutation group's signed score distribution while disrupting which action receives which score. The 35B control reaches 59.2% ID and 45% OOD, versus 69.2%/48% for SPICE (Appendix Table 4). This 10/3-point difference supports targeted feedback beyond the presence of an additional reward signal. Shuffling still outperforms outcome-only RL by 11.2/1 points, so correspondence does not explain the full gain or prove causal credit correctness. Separately, reduced solution summaries yield 67.2%/41%, testing reference completeness separately from action–score correspondence. These controls address complementary questions: the TIPS

comparison changes the scoring rule with reference content fixed, whereas shuffling disrupts action-specific feedback while preserving within-group score distributions.

**$\beta$ sensitivity.** Pilot calibration of $\beta$ to small, medium, and large shaping-reward ranges yields 67.2%, 69.2%, and 65.6% ID success for 35B, respectively, favoring medium shaping (Appendix B.1).

## 4 Related Work

**ML Development Agents.** MLAgentBench, MLE-bench, MLGym, and DeltaML-Bench provide executable evaluation (Huang et al., 2024; Chan et al., 2025; Nathani et al., 2025; Moukpe et al., 2026). AIDE and AIRA-dojo organize code search, while ARTS and Matryoshka structure diagnosis and exploration (Jiang et al., 2025; Toledo et al., 2025; Juneja et al., 2026; Qiang et al., 2026a). Related efforts develop interaction-trained agents (Liu et al., 2026b; Yang et al., 2026b) and scalable tasks and demonstrations in MLE-Dojo, MLE-Smith, SandMLE, and ML-AutoResearch (Qiang et al., 2025; 2026b; Zhou et al., 2026; Cai et al., 2026b). We complement these efforts with learnable diagnostic actions and privileged-context feedback within an existing scaffold.

**Tool Use Learning.** Toolformer, API-Bank, and ToolLLM study tool-selection and call-generation supervision (Schick et al., 2023; Li et al., 2023; Qin et al., 2024); ToolSandbox and $\tau$-bench evaluate dependencies across calls (Lu et al., 2025; Yao et al., 2025). ReTool and Tool-Star learn interaction strategies with outcome and hierarchical rewards (Feng et al., 2026a; Dong et al., 2026). We separately evaluate diagnostic access and tool-enabled supervision for evidence acquisition and use.

**Intermediate Feedback and Credit Assignment.** TIPS tracks correct-answer likelihood changes, and TACO uses before/after answer probes (Xie et al., 2026; Feng et al., 2026b). Agent0-VL combines verifier feedback with self-repair (Liu et al., 2026a); other methods use semantic role rewards or hindsight values (Xu et al., 2026; Tan et al., 2026). Self-Distilled RLVR uses positive likelihood-ratio weights without reversing advantage signs, while SEED uses hindsight skills for sampled-token learning (Yang et al., 2026a; Wu et al., 2026). SPICE instead adds signed action-support scores as turn rewards, retaining an executable outcome evaluator without a learned verifier (Appendix D.2).

## 5 Conclusion

MLToolBench and its training framework connect access to diagnostic evidence with learning to act on it. Guided SFT establishes tool-use behavior, and SPICE supplies action-level contextual support during RL. Across the evaluated synthetic suite, the full pipeline improves both Qwen backbones on ID and OOD tasks, and matched RL comparisons show higher success with SPICE than with outcome-only training. The results support diagnostic tool learning while leaving broader transfer and causal credit validation open.

## Limitations

Our evaluation covers 35 held-out synthetic tasks; transfer to natural ML pipelines remains untested. Ten rollouts per task measure fixed-checkpoint variability rather than independent RL training seeds, and aggregate gains do not establish consistency across individual tasks or sources, particularly for small OOD differences. The success threshold is defined in native metric units and is not scale-invariant, although all retained tasks passed our bound-based attainability check. Score shuffling supports the value of action–feedback correspondence without establishing causal credit correctness, and the contribution of the added scoring instruction is not isolated from that of the solution summary. Observation interventions also use each policy's own trajectories, mixing differences in tool invocation, visited states, and evidence use. Per-call reward calibration does not bound cumulative shaping, while frequency and diversity statistics cannot exclude rewards for redundant calls or changes to the outcome-only optimum. Finally, SPICE requires verified references and additional scoring passes, and equal raw SFT collection budgets do not ensure equal retained supervision; our comparisons therefore do not establish compute-matched superiority across training methods.

## Reproducibility Statement

Section 3.1 specifies the evaluation splits and success criterion. Appendix A summarizes task budgets, the success-rate metric, model checkpoints, and RL settings; the remaining appendices provide ablations, tools, and SFT details. The terminal success criterion, repeated-evaluation protocol, and outcome-only control are shared across the corresponding comparisons.

## AI Use Statement

Generative AI tools assisted literature search and summarization, exploration and refinement of research ideas and task designs, experimental design and interpretation of results, manuscript drafting and language editing, figure creation, and code development and debugging. Gemini 3.6 Flash was also used to generate training demonstrations, as described in Section 2.2. AI-assisted suggestions do not replace empirical validation. The authors retain responsibility for the final text, citations, methodological claims, code, figures, and reported results.

## References

Anthropic. Introducing Claude Sonnet 5, June 2026. URL https://www.anthropic.com/news/claude-sonnet-5.

Yuzhu Cai, Zexi Liu, Xinyu Zhu, Cheng Wang, Yanfeng Wang, and Siheng Chen. AceGRPO: Adaptive curriculum enhanced group relative policy optimization for autonomous machine learning engineering, 2026a. URL https://arxiv.org/abs/2602.07906.

Ziyang Cai, Amir Saeidi, and Harkirat Behl. ML-AutoResearch: Training Machine Learning Research Agents with Automatically Generated Environments, 2026b. URL https://arxiv.org/abs/2603.17216.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7e3767db483c942b883eb4f8cfb74e31-Abstract-Conference.html.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-Star: Empowering multi-tool collaborative web agent via reinforcement learning. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2026. doi: 10.1145/3805712.3809712. URL https://doi.org/10.1145/3805712.3809712.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4038c9208dfc22644c60ad39c24e5c53-Abstract-Conference.html.

Mingkuan Feng, Jinyang Wu, Hao Gu, Fangrui Lv, Ruihan Jin, Chuyuan Zhang, Zhengqi Wen, and Jianhua Tao. TACO: Tool-augmented credit optimization for agentic tool use, 2026b. URL https://arxiv.org/abs/2606.30251.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the large language model meets programming—the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024. URL https://arxiv.org/abs/2401.14196.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference*

*on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: Ai-driven exploration in the space of code, 2025. URL https://arxiv.org/abs/2502.13138.

Gurusha Juneja, Arnav Kumar Jain, Deepak Nathani, William Yang Wang, and Xin Eric Wang. Learning the ARTS of Search for Automated Discovery, 2026. URL https://arxiv.org/abs/2606.21891.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, 2023. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-VL: Exploring self-evolving agent for tool-integrated vision-language reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2511.19900. Oral presentation.

Zexi Liu, Jingyi Chai, Xinyu Zhu, Shuo Tang, Rui Ye, Weiyu Ma, Bo Zhang, Lei Bai, and Siheng Chen. ML-Agent: Reinforcing LLM agents for autonomous machine learning engineering. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306. PMLR, 2026b. URL https://openreview.net/pdf?id=kcPPWaoegr.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

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

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Josias Moukpe, Priyanka Aryal, and Matthew Kenney. DeltaML-Bench: Evaluating Machine Learning Agents on Real-World Research Repositories, 2026. URL https://arxiv.org/abs/2608.19653.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Mikhail Plekhanov, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Nicolaus Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A New Framework and Benchmark for Advancing AI Research Agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ryTr83DxRq.

Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander Alemi, and George Tucker. On variational bounds of mutual information. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5171–5180. PMLR, 2019. URL https://proceedings.mlr.press/v97/poole19a.html.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar V K, Rongzhi Zhang, ChangHao Li, Ian Wong, Sherry Yang, Percy Liang, Chao Zhang, and Bo Dai. MLE-Dojo: Interactive environments for empowering LLM agents in machine learning engineering. In *Advances in Neural Information*

*Processing Systems*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/0603c69125ad4b964bc9c4832f7b9f8f-Abstract-Datasets_and_Benchmarks_Track.html.

Rushi Qiang, Changhao Li, Haotian Sun, Yuchen Zhuang, Chao Zhang, and Bo Dai. Matryoshka Agent: Unfolding Sub-Agents for Long-Horizon Machine Learning Engineering, 2026a. URL https://arxiv.org/abs/2607.25090.

Rushi Qiang, Yuchen Zhuang, Anikait Singh, Percy Liang, Chao Zhang, Sherry Yang, and Bo Dai. MLE-Smith: Scaling MLE Tasks with Automated Multi-agent Pipeline. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0abcc0198bef043543d8837cfc1e18f7-Abstract-Conference.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon LLM agents, 2026. URL https://arxiv.org/abs/2603.08754.

The Microsoft AI Team. MAI-Thinking-1: Building a hill-climbing machine. Technical report, Microsoft AI, 2026. URL https://microsoft.ai/pdf/mai-thinking-1.pdf.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Lars Erik Saito Stenetorp, Nicola Cancedda, Jakob Foerster, and Yoram Bachrach. AI Research Agents for Machine Learning: Search, Exploration, and Generalization in MLE-bench. In *Advances in Neural Information Processing Systems*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/328b81881da145412f2bc56c998dfb6a-Abstract-Conference.html.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026. URL https://arxiv.org/abs/2607.14777.

Yutao Xie, Nathaniel Thomas, Nicklas Hansen, Yang Fu, Li Erran Li, and Xiaolong Wang. TIPS: Turn-level information-potential reward shaping for search-augmented LLMs. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fd881d3b625437354d4421818f81058f-Abstract-Conference.html.

Yuanda Xu, Zhengze Zhou, Hejian Sang, Xiaomin Li, Jiaxin Zhang, Xinchen Du, Sen Na, Zhipeng Wang, and Alborz Geramifard. TRIAGE: Role-typed credit assignment for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2606.32017.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled RLVR, 2026a. URL https://arxiv.org/abs/2604.03128.

Sherry Yang, Joy He-Yueya, and Percy Liang. Reinforcement Learning for Machine Learning Engineering Agents. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fe32a779aba1dc51a00861708944ee35-Abstract-Conference.html.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Jiayi Liu, Xiangjun Fan, Zhuokai Zhao, and Hong Yan. Synthetic sandbox for training machine learning engineering agents, 2026. URL https://arxiv.org/abs/2604.04872.

# A Experimental Protocol and Reproducibility

## A.1 Task Splits and Evaluation

SFT and RL use the same 80 training tasks; all 35 evaluation tasks are held out from demonstration collection and solution banks. Table 2 gives the collection and evaluation budgets. Each rollout ends after 50 agent steps or one hour, whichever occurs first. Original (standard MLAgentBench tools only) and + MLToolBench (the same tools plus diagnostics) share the same tasks and execution budgets. Each condition uses ten evaluation rollouts per task with the same fixed model or checkpoint. These are repeated task executions, not independent RL training runs.

| Partition | Instances | Runs per task | Total / purpose |
|---|---|---|---|
| Training | 80 | 100 | 8,000 raw demonstrations per SFT condition |
| ID evaluation | 25 | 10 | 250 evaluation runs per condition |
| OOD evaluation | 10 | 10 | 100 evaluation runs per condition |

Table 2: Collection and evaluation budgets. RL collects fresh on-policy rollouts.

Our success criterion is adapted from, but not identical to, MLAgentBench's relative-improvement criterion (Huang et al., 2024). Direct division by the baseline is undefined at zero and reverses the improvement direction for negative baselines. We instead compare the direction-adjusted absolute gain against a threshold based on the baseline magnitude, with a minimum absolute-gain floor to avoid counting negligible changes near zero. Let $m_{x,r}$ and $b_x$ be the final and starter metrics and $\sigma_x \in \{+1, -1\}$ indicate the favorable direction. Define

$$U_{x,r} = \mathbf{1}[V_{x,r} = 1 \wedge m_{x,r} \text{ finite} \wedge \sigma_x(m_{x,r} - b_x) > \max(0.10|b_x|, \delta)]\,. \tag{5}$$

We use $\delta = 0.01$ as the minimum absolute gain in the metric values returned by the task evaluator. The floor is defined on each evaluator's fixed native metric scale and is not scale-invariant. We checked the starter baselines and evaluator-defined metric bounds for all retained synthetic tasks; none has a success threshold rendered unattainable by these bounds. Valid zero and negative scores are retained. Non-finite final scores fail; a non-finite baseline requires repairing the evaluator configuration before evaluation. For $b_x > 0$ and $0.10b_x \geq \delta$, this reduces to the original strict 10% relative-gain rule. For $0 < b_x < 0.1$, the absolute floor makes our criterion stricter than a 10% relative-gain threshold. We therefore report results under this extended rule rather than claim exact equivalence to MLAgentBench preprocessing. We report

$$\mathrm{SR}(\mathcal{T}) = \frac{100\%}{10|\mathcal{T}|} \sum_{x \in \mathcal{T}} \sum_{r=1}^{10} U_{x,r}. \tag{6}$$

Each task has ten runs and equal weight. The observation intervention also includes all 250 ID rollouts per policy. Training-task test scripts compare submissions and starter baselines on the same partition and supply binary terminal rewards using the same criterion. SFT demonstration filtering is separate. Outcome rewards, shaped returns, and five-rollout tool coverage are separate measures.

## A.2 Model Checkpoints

The reported API model names are `gemini-3.6-flash` and `claude-sonnet-5`. Open checkpoints are `Qwen/Qwen3-8B`, `Qwen/Qwen3.5-35B-A3B`, and `deepseek-ai/deepseek-coder-33b-instruct`. Qwen3-8B is the official post-trained model; "unadapted" means before our task-specific SFT and RL. The coding baseline uses the pretrained `Qwen/Qwen2.5-Coder-14B` checkpoint.

## A.3 RL Configuration

RL starts from each backbone's tool-enabled full-SFT checkpoint. Table 3 lists its settings; reported results use the final checkpoint after 100 optimizer updates. SFT collection and optimization appear in Appendix D.1, prompts in Appendix E, and scoring details in Appendix D.2.

| RL configuration | Value |
|---|---|
| Optimizer updates / reported checkpoint | 100 / final step 100 |
| Rollout batch | 8 tasks $\times$ 8 rollouts = 64 trajectories |
| Optimizer / learning rate / scheduler | AdamW / $1 \times 10^{-6}$ / cosine |
| Minibatch size / update epochs / accumulation | 128 / 1 / 1 |
| Advantage estimation | Task-group mean centering; no standard-deviation scaling |
| Clip parameter / KL coefficient | 0.2 / 0 (KL disabled) |
| Terminal reward $R_i$ | Binary success under Equation 5 |
| Shaping calibration | Pilot mean absolute turn reward targeted to $[0.05, 0.3]$ (Appendix D.2) |
| Credit scorer | Same frozen rollout-policy snapshot in both branches |
| Privileged solutions | Up to 8 verified references per task; new solutions enter the next update |
| Training-time tool dropout (ablation) | 20%, sampled once at rollout start and fixed thereafter |
| Sampling temperature / top-$p$ / token limit | 1.0 / 1.0 / 1024 |
| Hardware / precision | 8$\times$ H200 / BF16 |

Table 3: RL configuration and shaping-scale calibration protocol. Rollout collection and optimization batch sizes are distinct. The calibration range refers to reward magnitude, not the coefficient.

# B Ablation and Behavioral Analysis

## B.1 Additional Ablation Results

Success rates use Equation 6, with 250 ID and 100 OOD runs per condition; the observation intervention reports ID only. Conditions shared with Table 1 use the same results.

**SFT ablations.** Figure 6(a,b) compares full SFT using demonstrations collected with or without MLToolBench, each starting from 8,000 raw trajectories before filtering and deduplication. Both policies are evaluated on Original and + MLToolBench, as defined in Section 3.1. The no-MLToolBench policy on Original provides a matched-interface baseline; the tool-trained policy on Original measures diagnostic removal. Tool-enabled SFT results reproduce Table 1.

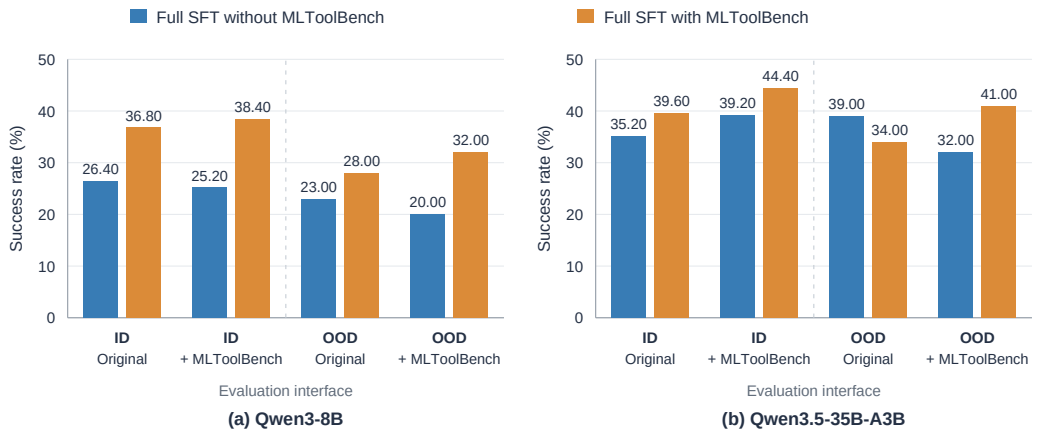



Figure 6: **Full-SFT training conditions and evaluation interfaces.** Each group compares SFT with and without MLToolBench demonstrations at a fixed evaluation interface. Original uses standard MLAgentBench tools without MLToolBench diagnostics; + MLToolBench adds these diagnostics at evaluation. Both panels share a scale and reproduce the tool-enabled SFT results in Table 1.

**RL and turn-level shaping.** Figure 7 compares full SFT with its RL continuation under MLToolBench; original-interface results remain in Table 1. The 8B outcome-only control ($\beta = 0$) reports 39.20% ID and 33.00% OOD success, versus 52.40% and 36.00% with SPICE. For each backbone, the three RL conditions share the SFT initialization, training data, tool interface, rollout budget, optimization settings, and binary outcome reward. Outcome-only disables shaping with $\beta = 0$, while TIPS and SPICE use their respective shaping methods. The 35B outcome-only control reports 48.00% ID and 44.00% OOD success, versus 69.20% and 48.00% with SPICE. RL with TIPS shaping achieves 44.40% ID (111/250) and 35.00% OOD (35/100) for 8B, and 55.20% ID (138/250) and 45.00% OOD (45/100) for 35B.

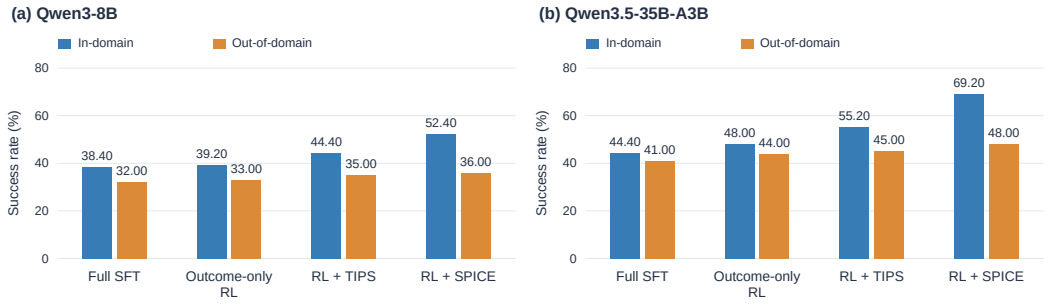



Figure 7: **RL-stage and reward comparisons with MLToolBench.** Full SFT and RL + SPICE values reproduce Table 1. Outcome-only RL ($\beta = 0$), RL + TIPS, and RL + SPICE are shown for both backbones. Gains over SFT measure the entire RL stage.

**Training-time tool dropout.** Dropping 20% of diagnostics once at each training rollout's start, with availability fixed thereafter, changes ID/OOD success from 52.4%/36% to 50%/31% for 8B and from 69.2%/48% to 58%/36% for 35B. Evaluation uses all tools: this tests incomplete training-time access, not robustness to missing tools at inference.

**$\beta$ sensitivity.** For Qwen3.5-35B-A3B with MLToolBench, we vary $\beta$ to calibrate pilot mean absolute turn-level shaping rewards to small ($[0.001, 0.05]$), medium ($[0.05, 0.3]$), and large ($[0.3, 1.0]$) ranges, obtaining ID success rates of 67.2%, 69.2%, and 65.6%, respectively. The medium range performs best, exceeding the alternatives by 2.0 and 3.6 percentage points. These intervals specify reward magnitudes, not $\beta$ values.

**Solution-summary completeness.** For 35B SPICE, the reduced-summary condition yields 168/250 ID successes and 41/100 OOD successes, compared with 173/250 and 48/100 using the full candidate summaries (Table 4). The reductions are 2.0 and 7.0 percentage points. Partial summaries retain an ID gain over outcome-only RL (67.2% versus 48.0%), but OOD success is lower (41% versus 44%). This comparison measures sensitivity to reference completeness; it is not a mismatched-reference or magnitude-matched control.

**Action–feedback correspondence.** We shuffle signed SPICE scores among actions within the same task and tool category (Diagnostic or Editing / Execution), preserving each permutation group's score distribution while disrupting its action correspondence. The matched RL control reaches 59.2% ID (148/250) and 45.0% OOD (45/100), versus 69.2%/48.0% for SPICE. Its remaining gain over outcome-only RL (11.2/1.0 points) indicates that correspondence does not explain all of the improvement. The comparison supports the value of targeted feedback, without proving that scores measure causal action contributions.

| 35B training condition | ID success (%) | OOD success (%) |
|---|---|---|
| Outcome-only RL | 48.00 (120/250) | 44.00 (44/100) |
| SPICE, full solution summaries | 69.20 (173/250) | 48.00 (48/100) |
| SPICE, reduced solution summaries | 67.20 (168/250) | 41.00 (41/100) |
| SPICE, shuffled scores | 59.20 (148/250) | 45.00 (45/100) |

Table 4: **Privileged-summary and score-assignment ablations.** Success rates for Qwen3.5-35B-A3B with MLToolBench. Shuffling permutes scores within tasks and tool categories. Full-summary SPICE reproduces the main result.

### B.2 Behavioral Analysis Details

Figure 8 reports tool-use coverage. For behavior-analysis tasks $\mathcal{B}$ and category $d$'s tool set $\mathcal{U}_d$, we compute

$$\text{Coverage}_d = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \sum_{r=1}^{5} \mathbf{1}[\exists s : \ \text{tool}(a_{x,r,s}) \in \mathcal{U}_d] \,. \tag{7}$$

Each of five complete rollouts contributes at most one count per category, regardless of repeated calls. Task-averaged values range from zero to five; 4.7 corresponds to 94% invocation coverage. This measures invocation; task success is evaluated separately with ten rollouts per task.

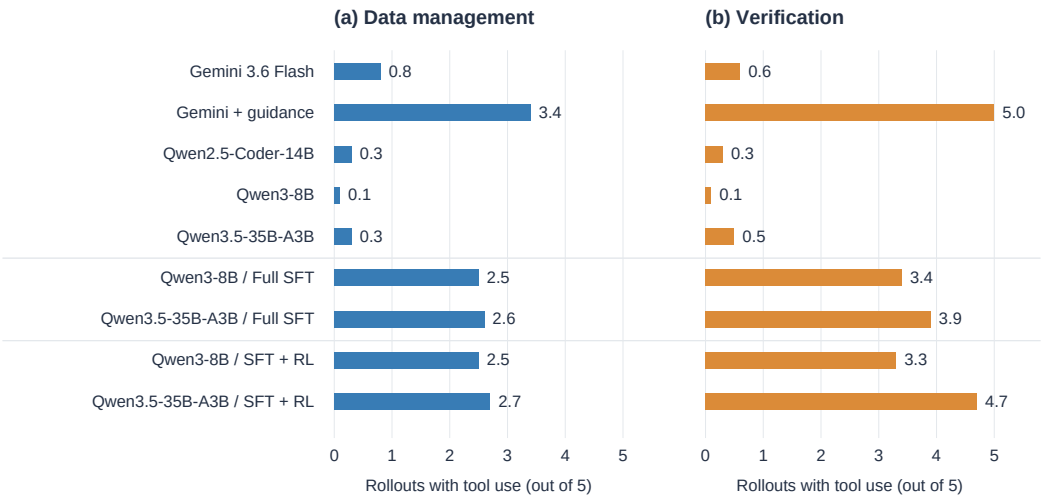



Figure 8: **Tool-use coverage by model and training condition.** Mean number of rollouts, out of five per task, invoking data-management or verification tools at least once. Separators distinguish unadapted baselines, full SFT, and full SFT + RL with SPICE.

**Call frequency and tool diversity.** Table 5 complements invocation coverage with mean calls and mean distinct tool IDs per rollout. Both count only MLToolBench tools, excluding editing/execution actions from the original MLAgentBench scaffold. For 35B, SPICE uses 11 calls and eight distinct tools on average, compared with nine/two for outcome-only RL and ten/three for TIPS. These statistics describe frequency and breadth; repeated calls can be useful after workspace changes, so they do not measure redundancy or rule out incentives from accumulated shaping rewards.

| Model / training | Coverage (out of 5) | | Mean per rollout | |
|---|---|---|---|---|
| | Data management | Verification | Calls | Distinct tools |
| 8B Full SFT | 2.5 | 3.4 | 5 | 2 |
| 8B RL + SPICE | 2.5 | 3.3 | 11 | 6 |
| 35B Full SFT | 2.6 | 3.9 | 7 | 4 |
| 35B Outcome-only RL | 0.8 | 1.3 | 9 | 2 |
| 35B RL + TIPS | 1.4 | 1.6 | 10 | 3 |
| 35B RL + SPICE | 2.7 | 4.7 | 11 | 8 |

Table 5: **MLToolBench invocation, frequency, and diversity.** Coverage counts rollouts invoking each category at least once; calls count invocations, and distinct tools count unique MLToolBench tool IDs within a rollout. Editing/execution actions are excluded.

### B.3 PAIRED DIAGNOSTIC-OBSERVATION INTERVENTION

**Evaluation setup.** We compare matched 35B outcome-only and SPICE policies on all 250 ID rollouts per policy (ten per task). Each policy generates its own trajectories. True is its unmodified evaluation, matching the corresponding main or RL-ablation result. Rollouts without diagnostic calls are also included in this evaluation.

**Intervention.** For a trajectory with MLToolBench diagnostic calls, we randomly select one read-only diagnostic return and branch from its saved history and workspace. True receives the original return. Masked preserves its response structure and execution status but marks measurements, conclusions, and derived suggestions unavailable. Paraphrased retains facts and values while changing presentation. Branches use paired seeds, identical remaining budgets and decoding settings; subsequent tools operate normally. Trajectories with no diagnostic call remain unchanged in all three arms and contribute their original outcomes. The task test script evaluates final success.

**Interpretation.** All three arms use the same 250-rollout denominator within each policy. Differences measure the effect of the observation intervention across the full ID evaluation, including unchanged no-call trajectories; they do not establish credit correctness or statistical significance.

| 35B training objective | Observation | ID success (%) |
|---|---|---|
| Outcome-only RL | True | 48.00 |
| | Masked | 46.40 |
| | Paraphrased | 47.60 |
| RL + SPICE | True | 69.20 |
| | Masked | 54.40 |
| | Paraphrased | 68.40 |

Table 6: **Diagnostic-observation intervention on ID tasks.** Both 35B policies retain all tools. All 250 ID rollouts are included. We randomly intervene on one diagnostic return where available; no-call trajectories retain their original outcomes in all arms.

# C MLTOOLBENCH INTERFACE DETAILS

MLToolBench contains **67 registered diagnostic tools**: seven shared tools plus 12 tabular, 15 time-series, 17 NLP, eight vision, and eight graph tools. These additions are separate from the scaffold's 17 base actions. Tables 7 and 8 give the complete inventory. The three functional groups in the main text describe purposes, whereas the five domain modules organize implementations; time-series tools also diagnose training curves from other domains.

**Interface and exposure.** Calls specify workspace files, logs, data fields, or model references and return measurements, flags, or heuristic suggestions. Registered tools are not all listed in every prompt: explicit tool selections or task-to-domain mappings determine prompt exposure, with additional diagnostics omitted by default. Prompt exposure does not itself enforce execution permissions.

**Execution and interpretation.** Most tools inspect existing data or artifacts. *Verify Hyperparam Effective* performs static analysis rather than verifying runtime values; *Trace Tensor Shapes* executes a dummy forward pass; *Estimate Peak Memory* briefly launches training and can inherit the script's side effects. Suggestions are not applied automatically. Heuristic scores are diagnostic proxies, not correctness guarantees, and model-dependent probes require the specified model or tokenizer. The tool registry does not itself enforce held-out-label isolation.

| Tool | Returned evidence / behavior |
|---|---|
| *Discover Log Metrics* | Metric names, sample values, and occurrence counts in logs or captured stdout. |
| *Extract Training Metric Series* | Value–step series for requested logged metrics. |
| *Verify Hyperparam Effective* | Static parameter-definition sites, conflicting values, and environment overrides. |
| *Estimate Peak Memory* | Observed peak GPU memory during a short training-script probe; OOM status. |
| *Trace Tensor Shapes* | Per-layer tensor shapes or failure traces from a CPU dummy forward pass. |
| *Check Config Consistency* | Rule-based checks of configuration relations and incompatible settings. |
| *Validate Output Format* | Record-format compliance, missing fields, and length or pattern violations. |

Table 7: Seven shared diagnostic tools. Names match the registered action interface.

| Tool | Returned evidence / behavior |
|---|---|
| **Tabular (12 tools)** | |
| *Detect Missing Values* | Per-column missingness and suggested imputation strategies. |
| *Identify Feature Types* | Inferred column types, unique counts, and ID/high-cardinality flags. |
| *Detect Label Imbalance* | Class counts, imbalance ratios, and resampling suggestions. |
| *Check Feature Normalization* | Numeric ranges, means, standard deviations, and scaling suggestions. |
| *Detect Outliers* | Per-column outlier counts and IQR bounds. |
| *Correlation Analysis* | Highly correlated feature pairs indicating possible redundancy. |
| *Cardinality Check* | Categorical unique counts and suggested encodings. |
| *Feature Skewness Checker* | Numeric-feature skewness and suggested transformations. |
| *Duplicate Row Detector* | Exact duplicate-row counts and proportions. |
| *Detect Train/Test Overlap* | Row overlap between two supplied datasets on selected columns. |
| *Encoding Strategy Advisor* | Encoding recommendations based on type, cardinality, and sample size. |
| *Feature Interaction Detector* | Numeric feature-product pairs ranked by estimated mutual-information gain. |

Continued on next page

Table 8 (continued)

| Tool | Returned evidence / behavior |
|---|---|
| **Time series (15 tools)** | |
| *Z-Score Anomaly Detector* | Global standardized outlier positions and magnitudes. |
| *IQR Anomaly Detector* | Outlier positions and interquartile-range bounds. |
| *Moving Window Anomaly Detector* | Locally unusual points relative to a moving window. |
| *Trend Detection* | Linear slope, fit statistics, and rank-based trend evidence. |
| *Plateau Detector* | Tail slope and relative change indicating a plateau. |
| *Convergence Estimator* | Early/late volatility and heuristic convergence indicators. |
| *Change Point Detection* | Largest detected mean-shift location and before/after means. |
| *Spike Detector* | Abrupt step-to-step changes relative to typical differences. |
| *Volatility Analysis* | Rolling and overall variability summaries. |
| *NaN/Inf Health Detector* | Non-finite values and large-magnitude warnings. |
| *Seasonality Detector* | Dominant frequency-derived period and periodicity strength. |
| *Autocorrelation Function (ACF)* | Autocorrelation by lag and significant-lag summaries. |
| *Partial Autocorrelation Function (PACF)* | Partial autocorrelation and suggested autoregressive order. |
| *Stationarity Test (ADF)* | ADF statistic, p-value, and stationarity/differencing suggestions. |
| *Granger Causality Test* | Lagged predictive-dependence tests between two supplied series. |
| **NLP (17 tools)** | |
| *Repetition Detector* | Exact and n-gram-based near-duplicate text ratios. |
| *Toxicity / Noise Filter* | Blank, markup, symbol-noise, and other heuristic text-quality flags. |
| *Language Consistency Checker* | Detected language proportions and mixed-language indicators. |
| *Instruction-Response Coherence* | Lexical-overlap proxy for instruction–response agreement. |
| *Token Length Distribution* | Tokenizer-based length statistics and potential truncation rates. |
| *Vocabulary Coverage Checker* | Unknown-token proportion for a supplied tokenizer. |
| *Prompt Template Validator* | Presence of expected prompt markers and compliance rates. |
| *Response Quality Scorer* | Heuristic length and repetition indicators for responses. |
| *Topic Distribution Analyzer* | TF-IDF clustering summaries and topic concentration. |
| *Difficulty Distribution Estimator* | Length/complexity-based difficulty proxies and their distribution. |
| *Data Source Mixing Analyzer* | Source counts, mixing ratios, and imbalance suggestions. |
| *Perplexity-based Outlier Detector* | Reference-LM perplexity statistics and high/low outliers. |
| *Reward Distribution Analyzer* | Reward-distribution statistics and discrimination indicators. |
| *KL Reference Divergence Estimator* | Estimated policy–reference KL on supplied text and penalty suggestions. |
| *Generation Diversity Checker* | Distinct n-grams, repeated outputs, and collapse indicators. |
| *Safety Content Detector* | Keyword-based sensitive-content flags and matched terms. |
| *Refusal Pattern Analyzer* | Refusal-phrase frequency and heuristic over-refusal indicators. |
| **Vision (8 tools)** | |
| *Class Balance Check* | Image-class counts and imbalance ratios. |
| *Image Normalization Stats* | Pixel ranges, channel means/standard deviations, and normalization suggestions. |
| *Image Shape Consistency* | Image-shape/channel distributions and mismatches. |
| *Duplicate Image Detector* | Exact and near-duplicate image counts using image hashes. |
| *Corrupt Image Detector* | Unreadable or corrupt image-file counts. |
| *Augmentation Auditor* | Static training-script scan for transforms and suggested additions. |
| *Train/Test Distribution Shift* | Differences in channel statistics between two supplied image sets. |
| *Per-Class Accuracy Breakdown* | Per-class accuracy and confusions from predictions and supplied labels. |
| **Graph (8 tools)** | |
| *Graph Size Summary* | Node/edge counts, density, feature dimensions, and split sizes. |
| *Degree Distribution* | Degree statistics, isolated nodes, and high-degree hubs. |
| *Node Label Balance* | Class counts on training nodes when available, otherwise supplied labels. |

Continued on next page

Table 8 (continued)

| Tool | Returned evidence / behavior |
|---|---|
| *Node Feature Health* | Feature scale, non-finite values, zero rows, and constant columns. |
| *Connectivity Components* | Connected-component counts and largest-component fraction. |
| *Self-loop & Duplicate Edge Detector* | Self-loop/duplicate-edge counts and reverse-edge coverage. |
| *Homophily Estimator* | Same-label edge fraction using supplied node labels. |
| *Split Integrity Check* | Split sizes, index overlaps, and supplied-label class coverage. |

Table 8: Complete domain-module inventory: 60 tools across five modules. Descriptions summarize implemented checks, including heuristic suggestions.

# D TRAINING DETAILS

## D.1 SUPERVISED FINE-TUNING

**Guidance construction and use.** The guidance-construction LLM receives the training task, its reference solution, and the synthetic environment generation mechanism. For each instance, it connects the prediction target, data modality, starter configuration, and generation conditions to concrete diagnostic checks. Guidance specifies the evidence to collect, the interpretation of alternative outcomes, and the corresponding edits and validation checks; different instances of the same source task can therefore receive different guidance. The resulting guidance is appended to the demonstration model's task prompt and specifies what to inspect and how the evidence should inform later decisions, without supplying the reference patch or fabricated observations. Student SFT inputs omit this guidance but retain the task, tool registry, actions, and actual tool outputs. Only assistant tokens receive next-token supervision, and the SFT checkpoint initializes RL.

**Collection and filtering.** Gemini 3.6 Flash generates 100 trajectories for each of 80 training tasks, yielding 8,000 raw demonstrations per condition. Retention requires successful execution, a valid submission, and improvement over a 10-seed starter baseline exceeding one baseline standard deviation in the favorable metric direction. This training-data filter is distinct from the terminal-success criterion; passing the SFT filter alone does not establish eligibility as a verified reference. We use embedding similarity to remove redundant trajectories, as in Agent0-VL (Liu et al., 2026a). Our selection rule retains the highest-improvement trajectory in each cluster. A 5% manual audit checks execution consistency, reasoning–observation agreement, and guidance leakage; failing examples are excluded. The reported retained corpus contains 6,700 trajectories. The full-SFT control without MLToolBench uses the same raw collection budget with Gemini 3.6 Flash using the original scaffold and no diagnostic guidance.

**Guidance examples.** Table 9 illustrates task-specific guidance grounded in starter models and data conditions. The ogbn-arxiv example follows the MLP-start schematic in Figure 2; it is distinct from the GraphSAGE-start configurations in the task manifest. Each example connects task characteristics to diagnostic evidence and conditional actions. These expanded examples summarize the guidance structure rather than reproduce complete collection prompts; an instance receives the checks relevant to its configuration.

**Optimization.** Table 10 records the SFT settings. Collection and deduplication are described above.

## D.2 RL WITH SPICE

**Token assignment and centering.** Equations 3–4 give the advantage estimator and policy objective. Only tool-call tokens enter the SPICE score, but every assistant token in a turn, including preceding reasoning, receives that turn's advantage. Tool observations are excluded from the loss. Each call emits one shaping reward; non-tool turns and invalid scoring contexts receive zero shaping. The shared baseline uses complete task-group returns and does not require trajectories to have aligned turns. Equal complete returns do not necessarily imply equal return-to-go at later turns, so zero group variance alone is not a reason to discard a group. With $\beta = 0$, the estimator becomes the outcome reward minus its task-group mean. We omit standard-deviation scaling while retaining per-trajectory token averaging, rather than adopting the full Dr. GRPO objective (Liu et al., 2025). Table 3 records training settings.

**Shaping-scale calibration.** Following the reward-scale calibration principle of TIPS (Xie et al., 2026), we use a short pilot on training tasks, separately for each backbone. The binary terminal reward fixes the reward scale at $c_R = 1$. For $N$ valid tool-use turns scored by the initial frozen snapshot, define

$$\widehat{m} = \frac{1}{N}\sum_{s=1}^{N} |d_s|, \qquad \beta = \frac{\kappa\, c_R}{\widehat{m} + \epsilon}, \quad \kappa \in [0.05, 0.3], \quad \epsilon = 10^{-8}. \tag{8}$$

Here $\kappa$ specifies the target mean absolute shaping reward relative to the terminal reward scale. Since $c_R = 1$, calibration targets $\frac{1}{N}\sum_s |\beta d_s| \in [0.05, 0.3]$ on pilot rollouts. This interval constrains reward magnitude, not $\beta$. Held-out evaluation tasks are excluded from calibration. This pilot target

| Task | Task-specific diagnostic guidance |
|---|---|
| `ogbn-arxiv` | **Context.** Improve paper-topic node classification from an MLP, following the diagnostic example in Figure 2. The presence of graph edges alone does not establish that replacing the MLP with GraphSAGE will improve performance. **Inspect.** Check node-feature ranges, per-column scales, and the active preprocessing pipeline. Compare training and validation curves, then assess whether graph connectivity and label homophily on permitted training labels motivate neighborhood aggregation. **Act on evidence.** If features have disproportionate scales or missing normalization, fit the transformation on permitted training data and re-run the MLP first. Compare the corrected MLP with the original baseline under the same split, evaluator, and budget. Only then test GraphSAGE if graph diagnostics motivate it; retain the same feature preprocessing and assess the additional gain from neighborhood information. Do not use hidden labels for diagnosis or treat a more complex architecture as an automatic improvement. |
| `cifar10` | **Context.** Improve ten-class image classification from a small CNN trained for five epochs; configured instances vary input scaling, training class balance, or cropping strength. **Inspect.** Check pixel ranges and channel statistics after preprocessing, inspect transformed images for retained object content, and compare training class counts with per-class validation performance. Read training and validation curves to distinguish optimization problems from overfitting. **Act on evidence.** If inputs retain an unintended integer scale, correct scaling and use training-derived normalization. If crops remove most object content, reduce cropping strength while preserving the expected input shape. If minority classes are underrepresented and perform poorly, test balanced sampling or class-weighted loss. If these checks are healthy, retain preprocessing and investigate optimization. Re-run under the same split and budget, comparing validation accuracy with the starter before accepting an edit. |

Table 9: Task-specific guidance examples for SFT demonstration collection. Checks and conditional actions reflect the task's starter, data conditions, and evaluation constraints; diagnostic observations must be obtained through actual tool calls.

| SFT configuration | Value |
|---|---|
| Adaptation | Full parameters |
| Epochs / optimizer | 3 / AdamW |
| Learning rate / scheduler / warmup | $1 \times 10^{-5}$ / cosine / 0 |
| Effective batch size | 64, using gradient accumulation |
| Maximum sequence length | 10,240 tokens |
| Hardware / precision | 8× H200 / BF16 |

Table 10: SFT optimization configuration.

does not guarantee the same magnitude throughout training and does not bound the total shaping reward accumulated over a trajectory.

**Privileged context and scoring.** References summarize verified demonstrations or successful solutions from earlier RL iterations of the same task. An agent writes one paragraph per candidate, describing its solution approach; the scoring context uses these summaries rather than full code or workspace diffs. TIPS and SPICE use the same candidate summaries. Hidden evaluator state and held-out labels are excluded. Sample up to eight references without replacement per task and update. Hold this subset and $q = \pi_{\text{old}}$ fixed during the update; newly verified solutions enter the next iteration. For each reference, append a training-only user message with a `SPICE_REFERENCE` block containing the candidate summary and a `SPICE_INSTRUCTION` block requesting assessment of the original next tool call. Teacher-force the same sampled call with and without this message, then average signed log-likelihood differences over references and action tokens. Neither branch sees the current tool observation or future trajectory. For context overflow, preserve history and action, truncating only the reference summary; use zero shaping if history and action alone exceed the limit. These scoring references are never shown to the acting student policy, including at evaluation; solution-informed guided Gemini is a separate reference condition.

**TIPS adaptation.** TIPS (Xie et al., 2026) uses the same agent-written candidate summaries as SPICE, treating them as reference answers rather than action-scoring context. We average token log-likelihoods within each summary, then average equally across candidates to define the potential. This differs from the original TIPS log probability of the valid-answer set. The turn signal is the post-interaction minus pre-interaction potential. A training-task pilot calibrates $\alpha$ to mean absolute scaled rewards in $[0.05, 0.3]$, matching SPICE's target magnitude range. Both methods share the acting-policy prompt, reference content, SFT initialization, rollout budget, and outcome evaluator, while differing in what the scoring model predicts.

**Interpretation and scoring cost.** Under a consistent joint distribution with positive probabilities,

$$\log \frac{p(a \mid h, c)}{p(a \mid h)} = \log \frac{p(c \mid h, a)}{p(c \mid h)}. \tag{9}$$

Different LLM prompt orders and length normalization need not preserve this identity. Poole et al. (2019) discuss the difficulty of conditional decoding for high-dimensional targets with high conditional entropy. This motivates our choice of local actions as shorter prediction targets; it does not establish a stability or accuracy guarantee for SPICE. The conditioned branches score $K|a|$ target tokens; total cost also includes reference prefill and the ordinary branch. Both directions allow teacher forcing. Unlike TIPS's temporal potential difference (Xie et al., 2026), SPICE compares conditioning contexts and has no corresponding policy-invariance guarantee. Scores measure action support, not causal contribution or subsequent use of evidence, and can alter the direction of policy updates compared with outcome-only GRPO.

# E SHARED AGENT PROMPTS

**Scope and provenance.** The following system prompt and task/action templates are used in our experiments. Following the organization of ML-Agent (Liu et al., 2026b), we separate task instructions, action descriptions, and response formatting. Placeholders are populated from each task environment and its current action registry. The supplementary `prompts/` directory contains the template text, and `prompt_config.json` maps experimental conditions to prompt components, tool interfaces, models, task splits, and training objectives.

## E.1 SHARED AGENT SYSTEM PROMPT

The shared prompt specifies the execution contract without prescribing a domain-specific diagnostic sequence. Both interfaces share this text and supply their respective action registries.

> You are an ML development agent working in an executable workspace. Improve the supplied starter solution for the stated task within the available budget and produce the required submission artifact.
>
> Use only the actions exposed in the current tool registry. Follow their names, argument schemas, and response protocol exactly. Do not invent an action or fabricate its observation. After issuing an action, use the environment's returned result to determine the next step.
>
> You may inspect files, edit permitted code, and execute experiments through available actions. Keep changes within the task's editable workspace. Preserve the evaluation protocol and respect the stated data-access and resource constraints.
>
> Report measured results with their metric and split. Distinguish observations from hypotheses and successful execution from achievement of the task criterion. Base the final report on artifacts and results that actually exist. When the budget is exhausted or the task is complete, use the registered completion action and identify the submission artifact and any unresolved execution issue.

## E.2 TASK AND ACTION-INTERFACE TEMPLATES

Task fields contain only information available to the acting policy. The baseline field identifies its metric and split; hidden evaluation scores and labels are excluded. The success rule specifies the task's metric direction and relative-improvement criterion.

```
TASK ID: {task_id}
TASK DOMAIN: {domain}
OBJECTIVE: {prediction_or_modeling_objective}
WORKSPACE: {workspace_root}
STARTER FILES: {starter_code_and_entry_points}
DATA: {accessible_data_paths_and_descriptions}
SPLIT POLICY: {training_validation_and_heldout_access_rules}
METRIC: {metric_name_and_optimization_direction}
BASELINE: {public_baseline_value_or_how_to_measure_it}
SUCCESS RULE: {task_specific_relative_gain_definition}
REQUIRED ARTIFACT: {submission_path_and_format}
EDIT CONSTRAINTS: {editable_files_and_protected_components}
RESOURCE BUDGET: {step_token_time_and_compute_limits}

Complete the task using the registered actions. The final held-out
assessment is performed by the evaluator under the split policy.
```

**Action-interface template.** The runtime serializes action descriptions, argument schemas, and return conventions from its current registry. Original includes standard MLAgentBench actions without MLToolBench diagnostics; + MLToolBench adds applicable diagnostics to those actions. Both use the runtime action and completion protocol.

```
The current action registry is supplied below. It is the complete
set of actions available in this run.

{serialized_current_action_registry}
```

```
Use this interaction format:
{runtime_action_serialization_and_completion_protocol}

Tool results are supplied by the environment after an action.
Do not write a tool result as though it were an observed response.
```

### E.3 Prompt Composition by Experimental Condition

| Condition | Agent-visible prompt | Training signal |
|---|---|---|
| Outcome-only RL | Shared system + task + registry + rollout history | Outcome reward only |
| RL + TIPS | Same agent prompt as outcome-only RL | Outcome + TIPS shaping |
| RL + SPICE | Same agent prompt as outcome-only RL | Outcome + SPICE shaping |
| SFT demonstration collection | Shared system + task + registry + history; guidance with MLToolBench | Full SFT on filtered trajectories |
| Unguided evaluation, both interfaces | Shared system + task + condition-specific registry + history | None |
| Guided Gemini 3.6 Flash evaluation | Shared system + task + registry + teacher guidance | None |

Table 11: Prompt composition used in the experiments. Outcome-only, TIPS, and SPICE share the acting-policy prompt; reward computation differs externally. Privileged scoring context is not appended to the acting policy's instructions.

SFT prompt construction is specified in Appendix D.1. For guided Gemini evaluation, guidance conveys core ideas considered in the solution while omitting full solution information and environment feedback. This solution-informed reference condition is distinct from ordinary unguided evaluation; student policies receive no such guidance at evaluation. At evaluation, the tool registry follows the evaluation condition rather than the demonstration source. For a fixed model, task, and guidance setting, Original and + MLToolBench share the system and task templates and differ in the available action registry; their interaction histories evolve independently.

**Configuration binding.** The supplementary configuration identifies guided and unguided evaluation, demonstration collection with and without MLToolBench, and outcome-only/TIPS/SPICE RL. Task membership is defined by `task_manifest.json`; the actor sees its opaque task ID rather than author-side fault metadata. Template files include content hashes for this export. Table 3 specifies RL settings, and Appendix D.1 specifies SFT filtering and optimization.

## F Task Manifest and Split Definitions

The supplementary `task_manifest.json` specifies the 115 instances used in this study, including their source links, task objectives, native metrics, starter descriptions, construction parameters, size limits, generation seeds, and partition policies. Table 13 indexes every instance. Each of the five domains—vision, NLP, tabular learning, time series, and graph learning—contains 16 training, five ID, and two OOD instances, giving 80/25/10 in total. Twenty training templates each contribute four configurations; changing only a rollout seed does not create a new task.

**ID and OOD at the task level.** ID instances reuse training sources, templates, and condition distributions with held-out supervised data or independently generated samples. OOD instances use ten source/target settings absent from training (Table 12); they test transfer within the same five broad domains, not to entirely unseen domains. Multiple configured instances can derive from one source, consistent with MLE-Smith and SandMLE (Qiang et al., 2026b; Zhou et al., 2026). All OOD settings are excluded from tool construction, SFT collection, RL training, and solution banks.

**Data partitions within tasks.** For training sources with independent or grouped examples, the protocol first allocates 60/40 meta-training/ID pools, then 70/15/15 model-fit/public-validation/hidden-test partitions within each pool, respecting grouping constraints. Time-series instances use chronological blocks and an embargo of at least lookback plus horizon; crossing windows are discarded. Node-classification ID is transductive: topology and unlabeled features can be shared, but supervised label pools are disjoint. CLRS uses separately generated graphs. OOD tasks use their recorded source-specific policies, including patient, notebook, or molecular-scaffold grouping. These partitions concern data inside an instance and are distinct from the 80/25/10 task split. Equation 6 and the task test scripts define success.

**Verification scope.** The manifest contains 115 unique instance IDs; all ID template IDs occur in training, and OOD template IDs and dataset labels are disjoint from training/ID. These checks concern the declared task configurations.

### F.1 OOD Sources and Transfer Axes

| Instance suffix | Source | Held-out source / target |
|---|---|---|
| cv-ood-01 | APTOS 2019 | Unseen medical-image source and ordinal target |
| cv-ood-02 | RSNA Breast Cancer | Unseen mammography source and probabilistic-F1 metric |
| nlp-ood-01 | AI4Code | Unseen code-notebook source and ranking target |
| nlp-ood-02 | BabyLM | Unseen language-modeling source and next-token target |
| tabular-ood-01 | California Housing | Unseen dataset and reconstruction target; downstream $R^2$ is secondary |
| tabular-ood-02 | Breast Cancer Wisconsin | Unseen clinical tabular source and calibration target |
| time_series-ood-01 | ECL | Unseen electricity-consumption source; forecasting target retained |
| time_series-ood-02 | G-Research Crypto | Unseen financial source and weighted-correlation objective |
| graph-ood-01 | MoleculeNet BBBP | Unseen molecular graphs and graph-level property prediction |
| graph-ood-02 | MoleculeNet BACE | Unseen molecular source and graph-level property prediction |

Table 12: OOD instances and the source or objective held out from training. Source links and exact construction parameters are included in the JSON manifest.

### F.2 Complete Instance Index

All suffixes below have the prefix `mltb-v1-`; `train`, `id`, and `ood` identify task membership. The seed is the registered generation seed, not a training seed or a repeated evaluation run. The JSON manifest provides each template's source, metric, construction parameters, and data partitions.

| Instance suffix | Template | Configuration | Seed |
|---|---|---|---|
| cv-train-01 | cv-cifar10 | healthy | 10000 |
| cv-train-02 | cv-cifar10 | input scale | 10001 |
| cv-train-03 | cv-cifar10 | rare class sampling | 10002 |
| cv-train-04 | cv-cifar10 | overaggressive crop | 10003 |
| cv-train-05 | cv-cifar100 | healthy | 10010 |
| cv-train-06 | cv-cifar100 | normalization scale | 10011 |
| cv-train-07 | cv-cifar100 | rare class sampling | 10012 |
| cv-train-08 | cv-cifar100 | overaggressive erasing | 10013 |
| cv-train-09 | cv-fashion | healthy | 10020 |
| cv-train-10 | cv-fashion | contrast scale | 10021 |
| cv-train-11 | cv-fashion | excessive shear | 10022 |

| Instance suffix | Template | Configuration | Seed |
|---|---|---|---|
| cv-train-12 | cv-fashion | excessive regularization | 10023 |
| cv-train-13 | cv-tgs | healthy | 10030 |
| cv-train-14 | cv-tgs | soft mask resize | 10031 |
| cv-train-15 | cv-tgs | wrong positive weight | 10032 |
| cv-train-16 | cv-tgs | overconservative threshold | 10033 |
| cv-id-01 | cv-cifar10 | input scale | 20000 |
| cv-id-02 | cv-cifar100 | rare class sampling | 20001 |
| cv-id-03 | cv-fashion | excessive regularization | 20002 |
| cv-id-04 | cv-tgs | soft mask resize | 20003 |
| cv-id-05 | cv-cifar10 | healthy | 20009 |
| cv-ood-01 | cv-aptos | overaggressive crop | 30000 |
| cv-ood-02 | cv-rsna | healthy | 30001 |
| nlp-train-01 | nlp-imdb | healthy | 11000 |
| nlp-train-02 | nlp-imdb | short context | 11001 |
| nlp-train-03 | nlp-imdb | unmasked pooling | 11002 |
| nlp-train-04 | nlp-imdb | class imbalance | 11003 |
| nlp-train-05 | nlp-spooky | healthy | 11010 |
| nlp-train-06 | nlp-spooky | vocabulary bottleneck | 11011 |
| nlp-train-07 | nlp-spooky | short context | 11012 |
| nlp-train-08 | nlp-spooky | overconfident probabilities | 11013 |
| nlp-train-09 | nlp-feedback | healthy | 11020 |
| nlp-train-10 | nlp-feedback | short context | 11021 |
| nlp-train-11 | nlp-feedback | missing target inverse | 11022 |
| nlp-train-12 | nlp-feedback | excessive regularization | 11023 |
| nlp-train-13 | nlp-textnorm | healthy | 11030 |
| nlp-train-14 | nlp-textnorm | case collapse | 11031 |
| nlp-train-15 | nlp-textnorm | lost sentence context | 11032 |
| nlp-train-16 | nlp-textnorm | rare type undersampling | 11033 |
| nlp-id-01 | nlp-imdb | short context | 21000 |
| nlp-id-02 | nlp-spooky | short context | 21001 |
| nlp-id-03 | nlp-feedback | excessive regularization | 21002 |
| nlp-id-04 | nlp-textnorm | case collapse | 21003 |
| nlp-id-05 | nlp-spooky | healthy | 21009 |
| nlp-ood-01 | nlp-ai4code | short context | 31000 |
| nlp-ood-02 | nlp-babylm | padding in loss | 31001 |
| tabular-train-01 | tab-house | healthy | 12000 |
| tabular-train-02 | tab-house | zero imputation | 12001 |
| tabular-train-03 | tab-house | integer category encoding | 12002 |
| tabular-train-04 | tab-house | missing target inverse | 12003 |
| tabular-train-05 | tab-spaceship | healthy | 12010 |
| tabular-train-06 | tab-spaceship | numeric missing sentinel | 12011 |
| tabular-train-07 | tab-spaceship | categorical missing collision | 12012 |
| tabular-train-08 | tab-spaceship | column order mismatch | 12013 |
| tabular-train-09 | tab-titanic | healthy | 12020 |
| tabular-train-10 | tab-titanic | unscaled features | 12021 |
| tabular-train-11 | tab-titanic | missingness sentinel | 12022 |
| tabular-train-12 | tab-titanic | excessive regularization | 12023 |
| tabular-train-13 | tab-nomad | healthy | 12030 |
| tabular-train-14 | tab-nomad | missing target inverse | 12031 |
| tabular-train-15 | tab-nomad | descriptor scaling | 12032 |
| tabular-train-16 | tab-nomad | overstrong regularization | 12033 |
| tabular-id-01 | tab-house | zero imputation | 22000 |
| tabular-id-02 | tab-spaceship | categorical missing collision | 22001 |
| tabular-id-03 | tab-titanic | excessive regularization | 22002 |
| tabular-id-04 | tab-nomad | missing target inverse | 22003 |
| tabular-id-05 | tab-titanic | healthy | 22009 |
| tabular-ood-01 | tab-california | global mean imputation | 32000 |
| tabular-ood-02 | tab-breast-calibration | overconfident probabilities | 32001 |
| time_series-train-01 | ts-etth-forecast | healthy | 13000 |
| time_series-train-02 | ts-etth-forecast | global scaler | 13001 |
| time_series-train-03 | ts-etth-forecast | output channel permutation | 13002 |
| time_series-train-04 | ts-etth-forecast | unnecessarily short context | 13003 |
| time_series-train-05 | ts-weather | healthy | 13010 |

| Instance suffix | Template | Configuration | Seed |
|---|---|---|---|
| time_series-train-06 | ts-weather | global scaler | 13011 |
| time_series-train-07 | ts-weather | output channel permutation | 13012 |
| time_series-train-08 | ts-weather | excessive regularization | 13013 |
| time_series-train-09 | ts-etth-impute | healthy | 13020 |
| time_series-train-10 | ts-etth-impute | wrong loss region | 13021 |
| time_series-train-11 | ts-etth-impute | missing mask input | 13022 |
| time_series-train-12 | ts-etth-impute | global scaler | 13023 |
| time_series-train-13 | ts-ventilator | healthy | 13030 |
| time_series-train-14 | ts-ventilator | wrong loss region | 13031 |
| time_series-train-15 | ts-ventilator | time order reversal | 13032 |
| time_series-train-16 | ts-ventilator | missing group resets | 13033 |
| time_series-id-01 | ts-etth-forecast | global scaler | 23000 |
| time_series-id-02 | ts-weather | output channel permutation | 23001 |
| time_series-id-03 | ts-etth-impute | global scaler | 23002 |
| time_series-id-04 | ts-ventilator | wrong loss region | 23003 |
| time_series-id-05 | ts-ventilator | healthy | 23009 |
| time_series-ood-01 | ts-ecl | global scaler | 33000 |
| time_series-ood-02 | ts-crypto | healthy | 33001 |
| graph-train-01 | graph-cora | healthy | 14000 |
| graph-train-02 | graph-cora | feature scale | 14001 |
| graph-train-03 | graph-cora | oversmoothing | 14002 |
| graph-train-04 | graph-cora | excessive regularization | 14003 |
| graph-train-05 | graph-citeseer | healthy | 14010 |
| graph-train-06 | graph-citeseer | missing self information | 14011 |
| graph-train-07 | graph-citeseer | feature scale | 14012 |
| graph-train-08 | graph-citeseer | excessive regularization | 14013 |
| graph-train-09 | graph-arxiv | healthy | 14020 |
| graph-train-10 | graph-arxiv | feature scale | 14021 |
| graph-train-11 | graph-arxiv | undersampled neighborhood | 14022 |
| graph-train-12 | graph-arxiv | excessive regularization | 14023 |
| graph-train-13 | graph-clrs | healthy | 14030 |
| graph-train-14 | graph-clrs | hint misalignment | 14031 |
| graph-train-15 | graph-clrs | padding in loss | 14032 |
| graph-train-16 | graph-clrs | insufficient message rounds | 14033 |
| graph-id-01 | graph-cora | feature scale | 24000 |
| graph-id-02 | graph-citeseer | feature scale | 24001 |
| graph-id-03 | graph-arxiv | excessive regularization | 24002 |
| graph-id-04 | graph-clrs | hint misalignment | 24003 |
| graph-id-05 | graph-cora | healthy | 24009 |
| graph-ood-01 | graph-bbbp | missing bond features | 34000 |
| graph-ood-02 | graph-bace | padding in pooling | 34001 |

Table 13: Complete 80/25/10 task-instance index. Condition names describe author-side configuration metadata and are not exposed in the actor prompt.