HARNESS POLICY DISTILLATION: ALIGNING AGENT POLICIES WITH RUNTIME SEMANTICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deploying a general-purpose model as an agent means running it inside a product-specific runtime harness that decides which actions are executable, how state advances, and when the loop may terminate. Ordinary trajectory supervision shows which action was taken but leaves the runtime semantics behind that action implicit. We introduce Harness Policy Distillation (HPD), which makes these runtime control semantics an explicit training target alongside actions to align the harness policy. HPD replays reference trajectories through the harness to recover the runtime semantics that are implicit in ordinary trajectory data, and represents them as control documents at each model-call boundary. These documents provide privileged information for a teacher to generate actions, while harness verification ensures that the resulting targets are executable under the runtime contract. We jointly trains a shared model on the runtime semantics recovered from control-state privileged information and native history, thereby internalizing these semantics into the model policy. At deployment, the trained model uses the original harness interface without requiring the control document or any additional supplement. On TRAJECT-Bench with the Pi harness, Qwen3-8B trained with HPD reaches 22.98% autonomous task success and 49.23% runtime compliance, against 3.09% and 19.90% for the unadapted model. Harness-verified teacher actions substantially improve the native deployment policy, while joint control-state supervision provides further gains in task completion and runtime compliance. HPD turns executable runtime semantics into dense supervision for learning harness-compatible actions at each decision boundary.

1 INTRODUCTION

A general-purpose model becomes a deployed agent only inside a product-specific runtime harness. The harness serializes visible history, exposes an approved action interface, enforces permissions and workflow stages, inserts tool observations, schedules subsequent model calls, and defines valid termination (Yang et al., 2024; Wang et al., 2025). For an organization adapting a model to its own product, the relevant target is therefore not task competence or tool syntax alone, but a harness-bound policy whose actions remain executable under that product's runtime contract.

Most post-training objectives supervise behavior without separately specifying the runtime control state on which that behavior depends. Demonstration and trajectory training maximize the likelihood of reference actions conditioned on observed histories, leaving the control state implicit in the mapping from history to action. Direct Preference Optimization (DPO) increases the relative likelihood of preferred outputs (Rafailov et al., 2023); Reinforcement learning with verifiable rewards (RLVR) optimizes expected return from executable checks (Shao et al., 2024; Guo et al., 2025), leaving the policy to infer reward-relevant control state through credit assignment. On-policy distillation (OPD) matches teacher behavior at student-generated prefixes (Agarwal et al., 2024), improving the coverage of states at which supervision is supplied while leaving the teacher's action distribution as the direct target. These objectives can learn runtime-sensitive behavior, but their standard forms leave the underlying control state to be inferred through action, preference, reward, or teacher supervision.

Within an agent, the model selects actions while the harness governs their execution. We distinguish two sources of policy mismatch between them. Supervision mismatch arises when the histories used for supervision differ from those generated by the agent's own rollout: a model may fit reference

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

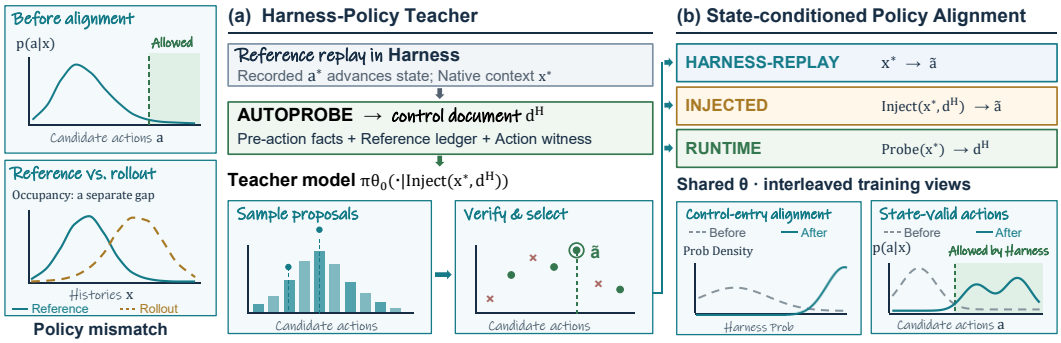


Figure 1: Harness Policy Distillation. Left: agent model–harness policy mismatches. (a) Recorded actions advance replay; AUTOPROBE compiles control documents; teacher-model proposals are verified and ranked by reference similarity. (b) Three views train one shared model for native action prediction, document-conditioned action prediction, and document reconstruction.

actions yet struggle to continue once its own actions lead to histories poorly covered during training (Ross et al., 2011; Bengio et al., 2015). Runtime-state mismatch arises when the model policy fails to reflect the harness's current execution requirements. Even on a valid reference history, action labels leave implicit which obligations remain and whether termination is allowed; the model may therefore repeat completed work or stop while required work is still pending. The two problems interact through the closed loop: each action changes the successor state and hence the context of the next decision. Improving the conditional policy at replayed boundaries is therefore a concrete training target, while autonomous task completion measures its effect across an entire execution. Because each action changes the next runtime state and model context, a local error can affect later decisions.

The harness provides a complementary source of supervision: its execution history and runtime bookkeeping identify the obligations and lifecycle conditions that govern the next action. For example, tool availability alone does not indicate whether the corresponding obligation has already been completed. Learning the next action from a reference trajectory leaves this distinction implicit; explicitly supervising the boundary's control state makes it a prediction. We therefore target this policy mismatch with explicit control-state and action supervision at replayed boundaries, and evaluate its effect through autonomous task completion.

To make this connection learnable, we propose **Harness Policy Distillation (HPD)**, which couples explicit control-state supervision with harness-verified action targets to align the model policy with the runtime semantics governing execution. At each replayed boundary, HPD compiles runtime observations and reference-derived control annotations into a privileged control document. This document guides a sampling model to propose actions, while harness verification restricts the training targets to candidates consistent with the execution and reference-control requirements. Recorded actions advance replay while selected candidates provide training targets, preserving the reference histories on which the paired supervision is defined.

To internalize these control semantics without requiring the document during autonomous execution, HPD jointly trains three views of one shared model. RUNTIME uses the compiled document as a target for control-state prediction from native history; INJECTED teaches action selection with these conditions made explicit. HARNESS-REPLAY trains the same verified action from native context alone, so action learning also matches the input available at deployment. Together, these views connect explicit control supervision to native action selection. The trained model acts through the original harness without an external control document or an additional model call (Figure 1).

2 RELATED WORK

Trajectory learning and policy distillation. Policies trained by behavior cloning run on histories induced by their own actions, motivating interactive imitation and scheduled sampling (Ross et al., 2011; Bengio et al., 2015). Sequence models instead fit state–action trajectories directly (Chen et al., 2021; Janner et al., 2021), while knowledge distillation and language-model post-training transfer teacher sequences, demonstrations, or preferences (Hinton et al., 2015; Kim & Rush, 2016; Ouyang

et al., 2022; Rafailov et al., 2023). On-policy distillation and agentic RL move supervision or reward toward student-generated prefixes (Agarwal et al., 2024; Shao et al., 2024; Guo et al., 2025). These methods primarily change the action target, reward, or visited occupancy; HPD instead adds runtime facts and reference-derived control labels as supervised targets at replayed decision boundaries.

Privileged and process supervision. Learning using privileged information introduces training-time variables that need not be available at inference (Vapnik & Vashist, 2009; Lopez-Paz et al., 2016). In self-distillation, privileged context lets a model supervise its counterpart without that context (Zhao et al., 2026a). π-Distill jointly trains a privileged-information-conditioned teacher and a student without that information using shared model parameters in multi-turn agentic environments (Penaloza et al., 2026). Process supervision identifies where a multistep solution becomes invalid (Uesato et al., 2022; Lightman et al., 2024). Training with Harnesses uses a harness-augmented current model as an on-policy self-teacher to internalize reasoning workflows (Zhao et al., 2026b). HPD pairs replay-derived control documents with actions selected by harness verification at the same decision boundary.

Tool agents and runtime harnesses. ReAct and tool-learning methods interleave language-model generation with environment actions or construct tool-use supervision (Yao et al., 2023; Schick et al., 2023; Yang et al., 2023; Qin et al., 2024). Interactive benchmarks make stateful execution central (Shridhar et al., 2021; Yao et al., 2022; Zhou et al., 2024; Liu et al., 2024; Mialon et al., 2024; Jimenez et al., 2024). SWE-agent and OpenHands further show that the agent interface, event stream, sandbox, and stopping rules are part of the deployed system (Yang et al., 2024; Wang et al., 2025). Evaluation work separates final task completion from intermediate progress and policy compliance (Ma et al., 2024; Yao et al., 2025; Lu et al., 2025). HPD uses this runtime lifecycle as a supervised object rather than treating it only as an external evaluator.

3 METHODOLOGY

3.1 CLOSED-LOOP EXECUTION UNDER A RUNTIME HARNESS

Let q denote a task instance, comprising the request and execution resources, and H the target runtime harness. Its native initialization produces the initial state s_0 from that instance. The state space $\mathcal{S}_H$ contains complete runtime configurations: execution history, tool interfaces, and the control fields maintained by H, including its permissions, scheduling state, obligation bookkeeping, and remaining budget. For replay, the fixture additionally supplies recorded tool-observation queues; their consumption positions are part of the state. Valid stopping states form $\mathcal{S}_H^{\mathrm{term}} \subseteq \mathcal{S}_H$. Add an absorbing failure state $\perp \notin \mathcal{S}_H$ for fatal contract violations or budget exhaustion, giving $\bar{\mathcal{S}}_H = \mathcal{S}_H \cup \{\perp\}$. Assume finite execution and let B be the first index with $s_B \in \mathcal{S}_H^{\mathrm{term}} \cup \{\perp\}$. For $b = 0, \ldots, B-1$, s_b is the complete state immediately before model call b, and x_b is its serialized model-visible context.

Let $\mathcal{X}_H$ be the space of native contexts and $\mathcal{A}$ the space of complete model outputs interpreted as actions, including tool calls and final responses. The harness exposes a serializer $\Omega_H : \mathcal{S}_H \to \mathcal{X}_H$ and a transition kernel:

$$x_b = \Omega_H(s_b), \qquad T_H : \bar{\mathcal{S}}_H \times \mathcal{A} \to \Delta(\bar{\mathcal{S}}_H), \tag{1}$$

where Δ denotes probability distributions over its argument space. $T_H(\cdot \mid s, a)$ is the successor law induced by executing action a from state s through the harness's parsing, permission checks, tool execution, observation insertion, scheduling, and stopping code. The successor is the next pre-call state or the terminal/failure outcome. Terminal and failure states are absorbing; recoverable errors become observations in nonterminal states. Thus T_H is supplied by the runtime implementation, while the *harness policy* denotes its state-dependent execution and stopping rules.

For a nonterminal state s, the immediately admissible action set is

$$\mathcal{A}_H(s) = \{a \in \mathcal{A} : T_H(\perp \mid s, a) = 0\}. \tag{2}$$

For deterministic replay, admissible actions reach a successor in $\mathcal{S}_H$. Reference-path agreement evaluates a sequence of decisions, while task success evaluates completed obligations, the answer, and termination. The model π_θ assigns sequence probabilities given a textual prompt, with trainable

parameters θ. Its harness action policy is $\pi_\theta(a \mid x)$ for $x \in \mathcal{X}_H$. It generates the closed loop from the harness-initialized state s_0:

$$x_b = \Omega_H(s_b), \qquad a_b \sim \pi_\theta(\cdot \mid x_b), \qquad s_{b+1} \sim T_H(\cdot \mid s_b, a_b). \tag{3}$$

For a trajectory $\tau = (s_0, x_0, a_0, \ldots, s_{B-1}, x_{B-1}, a_{B-1}, s_B)$ with $x_b = \Omega_H(s_b)$, conditioning on the task and initialization gives

$$p_{\theta,H}(\tau \mid q, s_0) = \prod_{b=0}^{B-1} \pi_\theta(a_b \mid \Omega_H(s_b)) T_H(s_{b+1} \mid s_b, a_b). \tag{4}$$

The executable agent is thus the process jointly induced by π_θ, Ω_H, and T_H. Each action determines a successor context, connecting boundary-level learning to closed-loop task completion.

3.2 Runtime semantic alignment gap

Policy distillation specifies both a context distribution μ and a teacher conditional $\pi^{\mathrm{T}}(\cdot \mid x)$ over actions. Here D_{KL} denotes forward Kullback–Leibler divergence, and $\mathbb{E}$ denotes averaging under the indicated distribution:

$$\mathcal{L}_{\mathrm{PD}}(\theta; \mu, \pi^{\mathrm{T}}) = \mathbb{E}_{x \sim \mu}\left[D_{\mathrm{KL}}\big(\pi^{\mathrm{T}}(\cdot \mid x) \,\|\, \pi_\theta(\cdot \mid x)\big)\right]. \tag{5}$$

For a fixed task collection, $\mu_H^\star$ is the empirical context distribution counting each replayed reference boundary once; $\mu_{\theta,H}$ normalizes the expected boundary counts under free rollout on that collection. Their discrepancy is the supervision mismatch; on-policy distillation moves supervision toward $\mu_{\theta,H}$.

Runtime-state mismatch concerns the conditional target at a given boundary. An action label specifies what was emitted; runtime control labels additionally identify lifecycle stage, remaining obligations, ready calls, and termination conditions. HPD supervises these labels together with a verified action at reference-history boundaries. Verification yields retained records; uniform sampling of these records defines the context distribution μ_H^{rep}. This filtering can change the relative weights of reference histories. Training fits the paired control and action targets there, and free rollout evaluates the resulting policy on its own successive decisions.

3.3 Harness Policy Distillation

HPD addresses this conditional-target gap by constructing paired control and action supervision at replayed reference boundaries (Figure 1). Reference actions advance the harness, while runtime observations and reference annotations are compiled into a privileged control document at each boundary. The frozen base model conditions on this document to propose actions, from which verification and near-reference selection produce the training target. The resulting context–document–action triples train one shared model through native-context action prediction, document-conditioned action prediction, and control-document reconstruction. At deployment, the model predicts actions directly from the native context.

Verified replay and control-document construction. Let $\tau^\star$ be a recorded task trajectory containing actions, tool observations, and reference stages; $a_b^\star$ is its action aligned to the b-th harness invocation. Superscript $\star$ marks reference replay, and superscript H marks harness-specific annotations. Starting from native initialization $s_0^\star$, replay submits each $a_b^\star$ to H, supplies recorded observations from per-tool queues, and captures the resulting $s_{b+1}^\star$ at the next boundary. The pre-action observation adapter $\mathrm{Observe}_H$ reads accessible runtime fields into snapshot o_b^H; the deterministic compiler Φ_H maps that snapshot and the reference annotations to a textual control document d_b^H. Each retained boundary satisfies

$$\begin{aligned} x_b^\star &= \Omega_H(s_b^\star), \qquad o_b^H = \mathrm{Observe}_H(s_b^\star), \\ d_b^H &= \Phi_H(o_b^H; \tau^\star, a_b^\star), \\ a_b^\star &\in \mathcal{A}_H(s_b^\star), \qquad T_H(s_{b+1}^\star \mid s_b^\star, a_b^\star) > 0. \end{aligned} \tag{6}$$

The transition condition checks the successor actually produced by replay; for deterministic replay its probability is one. AutoProbe comprises $\mathrm{Observe}_H$ and Φ_H. The document d_b^H contains observed runtime facts, reference-stage progress, and witness-derived labels for the next control, ready frontier,

and final permission. The latter fields provide privileged action information during training; field provenance and harness-specific adapters are detailed in Appendix B.

For illustration, specify a contract requiring `lookup(item_id=42)` before a final response, and a replay queue containing `value=7`. Initially lookup is pending and finalization is blocked. Executing the recorded call consumes that observation, completes the lookup obligation, and enables finalization. The documents label these pre-action states as `lookup=pending; final=blocked` and `lookup=complete; final=ready`. These transitions follow from the specified contract.

Harness-verified privileged teacher. Let θ_0 be the frozen base-model parameters. The prompt constructor $\mathrm{Inject}_H(x, d)$ supplies document d alongside native context x. The model samples K candidate actions $\tilde{a}_b^{(k)}$ per attempt, where k indexes candidates; the accepted set is $\mathcal{A}_b^{\mathrm{ver}}$ and its selected action is $\tilde{a}_b$:

$$\begin{aligned} \tilde{a}_b^{(k)} &\sim \pi_{\theta_0}\big(\cdot \mid \mathrm{Inject}_H(x_b^\star, d_b^H)\big), \qquad k \in \{1, \dots, K\}, \\ \mathcal{A}_b^{\mathrm{ver}} &= \Big\{ \tilde{a}_b^{(k)} : \; V_{H,b}(s_b^\star, \tilde{a}_b^{(k)}) = 1 \Big\}, \\ \tilde{a}_b &\in \mathcal{A}_b^{\mathrm{ver}}. \end{aligned} \tag{7}$$

$V_{H,b}(s, a) \in \{0, 1\}$ checks admissibility, reference-stage consistency, and exact agreement with the recorded control decision, tool names, and complete normalized tool arguments. We select one verified candidate by cosine similarity to the reference over its remaining fields. The recorded action $a_b^\star$ advances replay, while the selected $\tilde{a}_b$ supplies the training target. Appendix B.3 details parameter replay, candidate selection, corrective resampling, and reference-based continuation during offline construction. Each retained record is the triple $(x_b^\star, d_b^H, \tilde{a}_b)$.

Three-view training and native deployment. Let d range over control-document strings. The constructor $\mathrm{Probe}_H(x)$ requests document prediction from native history without supplying the target document. Superscripts T, I, and R denote the three prompt contracts of one shared model:

$$\begin{aligned} \pi_\theta^T(a \mid x) &= \pi_\theta(a \mid x), \\ \pi_\theta^I(a \mid x, d) &= \pi_\theta(a \mid \mathrm{Inject}_H(x, d)), \\ \pi_\theta^R(d \mid x) &= \pi_\theta(d \mid \mathrm{Probe}_H(x)). \end{aligned} \tag{8}$$

HARNESS-REPLAY (T) predicts $\tilde{a}_b$ from the native context; INJECTED (I) predicts the same action with d_b^H supplied; RUNTIME (R) predicts d_b^H from native history. These views connect history to control, control to action, and history directly to the deployed action through shared parameters. The paired document–action records define a joint teacher: R fits its document marginal, I its action conditional given the document, and T its native-runtime action marginal. By the KL chain rule, the R and I losses decompose joint-teacher matching; T directly supervises the deployment policy. Let $\mathcal{L}_T$, $\mathcal{L}_I$, and $\mathcal{L}_R$ be these conditional teacher-to-student KL losses averaged over retained records (Appendix F). Their weighted objective is

$$\mathcal{L}_{\mathrm{HPD}} = \alpha_T \mathcal{L}_T + \alpha_I \mathcal{L}_I + \alpha_R \mathcal{L}_R, \qquad \alpha_T + \alpha_I + \alpha_R = 1. \tag{9}$$

Here $\alpha_v \geq 0$ is the supervision frequency for view $v \in \{T, I, R\}$.

Sequence-level distillation and implementation. For each view, let c be its prompt from Equation 8 and y its stored completion: $\tilde{a}_b$ for T/I or d_b^H for R. The point mass δ_y assigns probability one to y, including its completion boundary. With $|y|$ tokens, token y_j, and target prefix $y_{<j}$, autoregressive factorization gives

$$D_{\mathrm{KL}}(\delta_y \,\|\, \pi_\theta(\cdot \mid c)) = -\log \pi_\theta(y \mid c) = -\sum_{j=1}^{|y|} \log \pi_\theta(y_j \mid c, y_{<j}). \tag{10}$$

Teacher-forced token prediction thus implements sequence-level forward KL for each hard target (Kim & Rush, 2016). Averaging per-record NLLs recovers each view's conditional teacher KL up to a constant teacher entropy; Appendix F derives this aggregation for shared prompts. The implemented loss $\ell_\theta(y, c) = -\log \pi_\theta(y \mid c)/|y|$ weights each sequence KL by inverse completion length, putting documents, final responses, and tool-call waves on a per-token scale. Training uses stored targets and cycles through view-specific batches. Deployment uses only $\pi_\theta^T(\cdot \mid x)$; the full implemented objective is given in Appendix F.

4 EXPERIMENTS

4.1 SETUP

We evaluate Qwen3-8B (Yang et al., 2025) with LoRA adaptation (Hu et al., 2022) on 583 held-out TRAJECT-Bench tasks (He et al., 2026). Table 1 compares HPD with the base model and supervised, preference, reinforcement-learning, and distillation baselines. We report task success (TS), final-answer F1, gold-history action accuracy (Action-Acc), reference-prefix success (Step-SR), runtime compliance (RC), and exact-path success (Exact), all produced by `pi-runtime`. We then test component and schedule effects, boundary behavior, adaptation across models and harnesses, and transfer to a new task collection. Appendices C and E specify the training, baseline implementations, seed study, and metric definitions.

4.2 MAIN COMPARISON

Table 1: Qwen3-8B comparison on TRAJECT-Bench/Pi. Scores (%) are from `pi-runtime`; TS, RC, and Exact use 583 cases. HPD-T and HPD share document-conditioned, verifier-selected teacher targets; HPD-T trains only the Harness-Replay (T) view, while HPD adds I/R. Baselines are detailed in Appendix C.

Method	Supervision source	TS	F1	Step-SR	RC	Exact	Action-Acc
Base	—	3.09	26.82	26.45	19.90	2.57	53.41
SFT	Ground truth	2.06	2.40	47.20	55.92	1.72	2.06
DPO	Preference labels	7.20	33.62	11.12	25.56	5.15	**72.38**
GiGPO	Task reward	4.46	29.65	26.17	21.61	3.43	54.63
GRPO	Task reward	3.43	27.59	25.79	20.41	2.74	54.35
GSPO	Task reward	7.72	37.61	13.46	28.64	6.69	54.30
GDPO	Task reward	4.63	35.04	19.07	23.50	3.77	54.11
OPD	Teacher model	15.61	41.04	57.76	**66.21**	14.07	59.91
OPSD	Privileged info	13.55	26.27	55.98	26.93	10.98	63.97
OPHSD	Teacher + harness	3.09	29.45	27.66	19.73	2.06	53.22
TurnOPD	Teacher model	9.43	29.02	50.70	44.77	6.52	59.77
HPD-T	Harness runtime	20.58	39.96	59.25	43.74	15.61	61.40
HPD	**Harness runtime**	**22.98**	**41.22**	**63.79**	49.23	**18.70**	64.95

Table 1 separates local ranking (Action-Acc), execution (Step-SR and RC), answer overlap (F1), and completion (TS and Exact). HPD-T reaches 20.58 TS, above the strongest baseline, OPD at 15.61. Full HPD leads in TS, F1, Step-SR, and Exact: its 22.98 TS is 43 completed tasks above OPD (7.38 points; 47.3% relative). The five-seed paired analysis supports both TS differences (Appendix C.5).

Reference supervision: SFT and DPO. SFT raises RC from 19.90 to 55.92 and Step-SR from 26.45 to 47.20, yet lowers TS from 3.09 to 2.06 and F1 from 26.82 to 2.40. Fitting recorded gold actions improves reference-path metrics here without preserving answer quality or task completion under Pi deployment, illustrating the mismatch introduced in the Introduction. DPO achieves the best Action-Acc alongside 11.12 Step-SR and 7.20 TS. Its advantage is local reference-action ranking; its rollout follows the reference prefix less often. RC separately measures compliance with the execution contract. Together, these metrics distinguish fitting the next reference action, maintaining its reference path, and completing an executable task.

Task rewards: GRPO and its variants. GiGPO and GRPO give modest TS gains (4.46 and 3.43), with Step-SR (26.17 and 25.79) close to Base. GSPO leads the reward-based baselines in TS and F1 (7.72 and 37.61); GDPO reaches 35.04 F1 but only 4.63 TS. Their Step-SR values (13.46 and 19.07) show that higher answer overlap need not sustain the reference path.

Model teachers: self-distillation and OPD. OPHSD leaves TS at 3.09. OPSD reaches 13.55 TS and 55.98 Step-SR with F1 near Base (26.27 versus 26.82), showing stronger execution without higher answer overlap. OPD achieves the highest RC (66.21), with 57.76 Step-SR, 15.61 TS, and 14.07 Exact at 41.04 F1. TurnOPD also improves reference-prefix execution and compliance over Base, reaching 50.70 Step-SR, 44.77 RC, 9.43 TS, and 6.52 Exact. OPD has the highest RC, while HPD's higher TS shows that compliance and completed tasks rank differently.

Harness-verified replay and runtime semantics. HPD-T uses the same document-conditioned, verifier-selected teacher as Full HPD, but trains only the native replay-action view. Relative to OPD, it gains 4.97 TS points and 1.54 Exact points while its mean F1 is slightly lower (39.96 versus 41.04). The verified-target and native-action pipeline thus improves closed-loop completion. Adding I/R training views raises TS by another 2.40 points, Step-SR by 4.54 points, and Exact by 3.09 points. The runtime semantics injection and control-document views contribute to this second gain.

4.3 Cross-benchmark robustness study

We evaluate the TRAJECT-Bench-trained checkpoints on 92 BFCL v4 cases (Patil et al., 2025) without BFCL-specific training. BFCL supplies new tasks and default tool execution; the Pi agent, checkpoints, and model-side settings remain fixed (Appendix C.6). Full HPD reaches 21.74 TS, and 48.12 Step-SR versus Base's 13.04, and 42.03. HPD-T reaches 13.04 TS, and 46.16 Step-SR: adding I/R raises TS by 8.70 and Step-SR by 1.96 points.

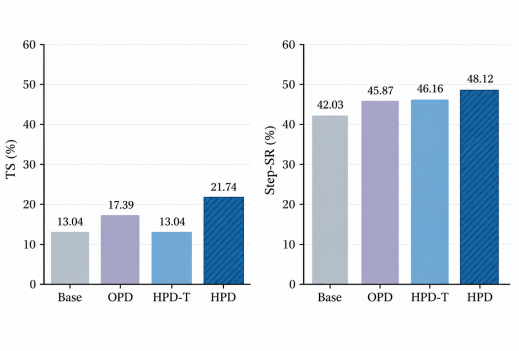


Figure 2: Zero-shot transfer to BFCL v4 from above checkpoints without specific training.

HPD-T improves answer overlap and prefix execution over Base but not task completion. On these new tasks, Full HPD's completion gain is consistent with I/R connecting replay actions to Pi's pending obligations and termination beyond source-task patterns. When the test tasks extend beyond the range of trajectories encountered during training, the performance gains of HPD-T drop significantly; in contrast, HPD reinforces the alignment of the harness policy—rather than overfitting to a specific task distribution—by leveraging runtime semantic training (I/R), which accounts for its superior generalization capabilities.

4.4 HPD components and training schedule

Table 2 compares the replay-action view and control-document views at matched exposure: each active view sees 20k training batches. Joint variants interleave view updates, whereas staged variants apply 20k batches per view in consecutive phases. Appendix C.2 explains the bookkeeping.

Table 2: Qwen3-8B HPD component and schedule ablation with 20k training batches per active view. T is the replay-action view; I and R are the injected- and runtime-document views. Arrows denote consecutive phases. Scores are percentages from `pi-runtime`.

HPD variant	TS	F1	Step-SR	RC	Exact	Action-Acc
HPD-T	20.58	39.96	59.25	43.74	15.61	61.40
T→I (staged)	18.01	40.23	47.99	34.99	14.24	64.07
T→R (staged)	7.03	17.47	37.29	18.35	5.15	56.64
T+I joint (without R)	20.58	39.66	56.92	41.68	14.92	62.90
T+R joint (without I)	20.58	40.63	63.13	48.03	16.81	62.80
Full HPD (T+I+R joint)	**22.98**	**41.22**	**63.79**	**49.23**	**18.70**	**64.95**

Using HPD-T as the anchor, the two-view rows distinguish the contributions of I and R before combining all three views. T+R improves Step-SR from 59.25 to 63.13 and RC from 43.74 to 48.03 while TS stays at 20.58; T+I also leaves TS unchanged and lowers both execution metrics. Only the joint T/I/R run raises TS beyond HPD-T, suggesting that I and R contribute to completion in combination rather than as independent additions.

Training order matters: T→R drops TS from 20.58 to 7.03 and RC from 43.74 to 18.35; T→I also degrades closed-loop performance. At matched updates and per-view exposure, both joint pairs outperform their staged counterparts on TS and RC. Interleaving retains replay-action supervision throughout training, while staging ends on an auxiliary target.

4.5 TOKEN DISTRIBUTIONS AND STATE-CONDITIONED ACTION ALIGNMENT

At the same replayed decision boundaries, we examine three aspects of action formation: confidence in the reference control entry, uncertainty over the reference-conditioned continuation, and validity of the action generated without a reference prefix. Figure 3 compares Base, SFT, GRPO, and HPD; Appendix E.1 defines the measurements.

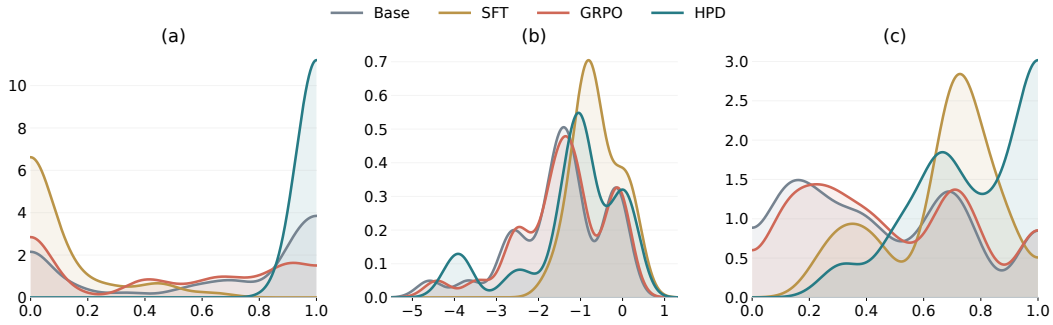


Figure 3: **Control entry and reference-conditioned continuation.** At identical harness states: (a) probability of the reference harness control token; (b) $\log_{10}$ of mean full-vocabulary predictive entropy in bits; (c) geometric mean reference-token probability. Continuation statistics use reference prefixes, exclude the initial control token, and cover at most 63 positions. KDEs share a bandwidth across models within each panel and integrate to one in the plotted coordinate.

In panel (a), HPD assigns a mean probability of 0.9962 to the reference control entry, compared with 0.6242 for Base, 0.5087 for GRPO, and 0.1296 for SFT. This is alignment with the reference entry at a supplied boundary; SFT can score higher on rollout compliance while assigning less probability to that particular entry.

Panel (b) examines the distribution of continuation entropy across those boundaries. SFT's density shifts toward higher entropy and loses much of the low-entropy mass visible for Base. HPD and GRPO retain substantial overlap with Base, even as HPD's reference control-entry confidence rises sharply. HPD's stronger execution alignment thus coexists with a broadly overlapping reference-conditioned entropy profile, rather than a uniform concentration of continuation predictions.

In panel (c), SFT and HPD assign similar median probabilities to reference continuation tokens under the correct prefixes (0.7038 and 0.6966). When they generate complete actions from their own prefixes, however, the state-validity rates in this diagnostic are 7.40% and 81.48%, respectively. An action couples its control entry to a continuation generated from the model's own earlier tokens. Neither the entry score in panel (a) nor a reference-conditioned continuation score tests that complete action. HPD's T view fits verifier-approved actions, while I/R tie those actions to the runtime state during training. The contrast echoes the main comparison: SFT's Step-SR and RC improve, but F1 and TS fall. Free rollout tests whether this state–action link holds across successive decisions.

4.6 ADAPTATION ACROSS ARCHITECTURES AND HARNESSES

Figure 4 shows higher TS and F1 across all evaluated architectures and harnesses after separate adaptation. Each within-study Base–HPD pair measures the effect of applying the same construction and training procedure to that configuration. Llama3-8B gains 20.07 TS points (Base TS: 0), while Gemma4-12B (Gemma Team et al., 2026) gains 16.47 points and Qwen3-30B-A3B (Yang et al., 2025) gains 12.69 points. Separate adapters for ReAct, Compiler, and OpenCode gain 9.44–18.35 TS points across different scheduling, dependency, and permission semantics. Each adaptation rebuilds replay, control documents, and verification for its harness while retaining the T/I/R objective and native-runtime deployment. The two axes stress different parts of HPD: the backbone comparison changes the policy under Pi, whereas the harness comparison changes the execution contract under Qwen3-8B. In each setting, replay and verification construct compatible action targets, and T/I/R connects them to native execution. The consistent gains support portability of this construction across both model and runtime changes.

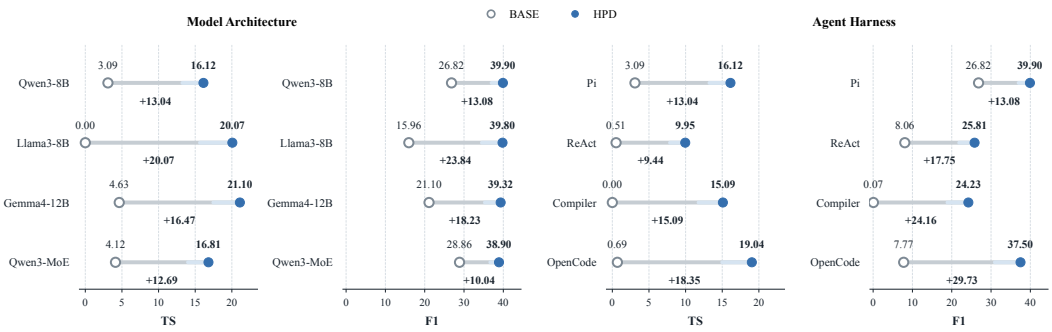


Figure 4: TS and F1 across model architectures (left two panels) and separately adapted harnesses (right two panels). All scores are percentages. The Qwen3-8B/Pi Base scores match Table 1; its HPD scores use the 20k-optimizer-step checkpoint.

4.7 API MODEL DEPLOYMENT REFERENCES

Figure 5 places HPD beside four API-accessed models under the same Pi agent, TRAJECT-Bench tasks, and evaluation settings. HPD reaches 22.98 TS versus 22.30 for DeepSeek V4 Flash, while GPT-5.4-mini reaches 41.92 F1 versus 41.22 for HPD. HPD uses the same checkpoint as in Table 1; the API rows replace only the model backend. The different TS and F1 rankings are consistent with HPD's training target: verifier-selected actions and control documents supervise the obligation and termination behavior that TS measures alongside answer quality.

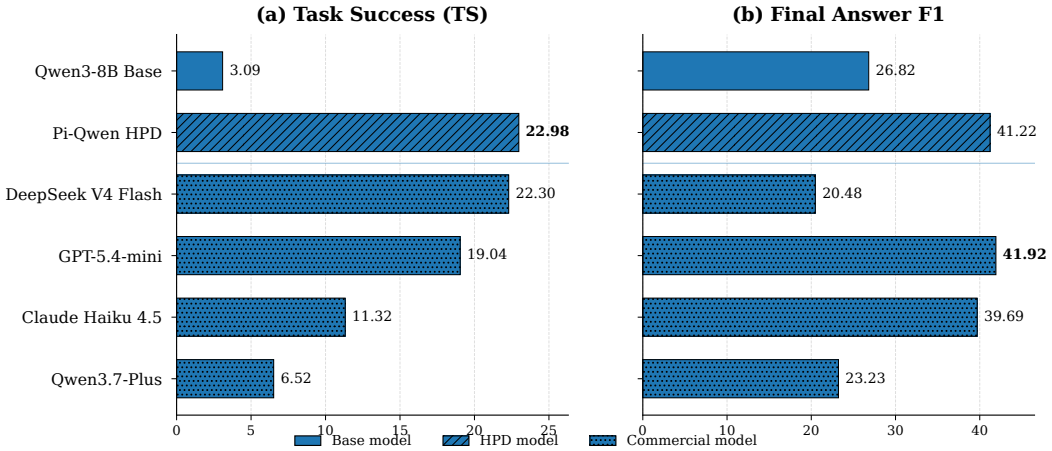


Figure 5: Task success and final-answer F1 for the Qwen3-8B Base model, Pi-Qwen HPD, and API-accessed models in the same Pi agent and evaluation setup. Values are percentages.

5 CONCLUSION

Our study addresses policy mismatch between the model and harness within an agent by targeting *runtime-state mismatch*. It teaches the model both what to do and which execution conditions to account for. HPD turns executable runtime contracts into verified action targets and explicit control supervision and three shared-parameter views T, I, R, connect them to the native deployment policy. On TRAJECT-Bench with Pi, HPD reaches 22.98 task success versus OPD's 15.61 at nearly equal aggregate F1. HPD-T already benefits from verified action supervision, while joint control-state training further improves completion. This benefit extends to new tasks: on 92 BFCL v4 cases without retraining, Full HPD reaches 21.74% task success versus HPD-T's 13.04%. Separate adaptations improve performance across model backbones and harnesses. Deployment retains a single model and the existing agent interface, without an external control document. Together, these results support aligning agent policies not only with task actions, but also with the runtime semantics that govern execution.

Reproducibility Statement

Appendix B specifies replay assumptions and document provenance; Appendix C records the data split, training configuration, and evaluation protocol; Appendix E defines the metrics; and Appendix F derives the training objective.

Ethics Statement

HPD optimizes compliance with a specified runtime contract. Responsible deployment therefore pairs model adaptation with an audited contract, externally enforced permissions, tool isolation, and execution monitoring. The TRAJECT-Bench experiments use replayed tool outputs and argument-key checks; live deployment additionally calls for validation of tool effects and argument values. These responsibilities remain part of the product runtime throughout adaptation and deployment.

AI Use Statement

Generative AI tools assisted with drafting and revising the manuscript, literature retrieval, research ideation, review of the methodology and interpretation of results, mathematical derivation and notation checks, reference formatting, and visualization code for reported results. The frozen language model used to construct training targets is part of the method described in Section 3.3. The authors reviewed the AI-assisted work and take responsibility for the paper's text, claims, references, results, and artifacts.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, et al. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html.

Lili Chen, Kevin Lu, Aravind Rajeswaran, et al. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Gemma Team et al. Gemma 4 technical report. arXiv preprint arXiv:2607.02770, 2026. URL https://arxiv.org/abs/2607.02770.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. TRAJECT-Bench: A trajectory-aware benchmark for evaluating agentic tool use. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TZWnWvsQ0X.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015. URL https://arxiv.org/abs/1503.02531. Presented at the NIPS 2014 Deep Learning Workshop.

Edward J. Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, et al. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327. Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Hunter Lightman, Vineet Kosaraju, Yura Burda, et al. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Shih-Yang Liu, Xin Dong, Ximing Lu, et al. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Y1rHxA58If.

Xiao Liu, Hao Yu, Hanchen Zhang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

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, et al. 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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Chang Ma, Junlei Zhang, Zhihao Zhu, et al. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, et al. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25ae35b5b1738d80f1f03a8713e405ec-Abstract-Conference.html.

Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.04942.

Yujia Qin, Shihao Liang, Yining Ye, et al. 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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, et al. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, et al. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Jonathan Uesato, Nate Kushman, Ramana Kumar, et al. Solving math word problems with process- and outcome-based feedback. arXiv preprint arXiv:2211.14275, 2022. URL https://arxiv.org/abs/2211.14275. Presented at the NeurIPS 2022 MATH-AI Workshop.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5–6):544–557, 2009. doi: 10.1016/j.neunet.2009.06.042.

Xingyao Wang, Boxuan Li, Yufan Song, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a4b6ad6b48850c0c331d1259fc66a69c-Abstract-Conference.html.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, et al. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Rui Yang, Lin Song, Yanwei Li, et al. GPT4Tools: Teaching large language model to use tools via self-instruction. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/e3936777793767624f2821cec8bdd02f1-Abstract-Conference.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, et al. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-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.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=Jpxfof0EaS.

Zhengyang Zhao, Lu Ma, and Wentao Zhang. Training with harnesses: On-policy harness self-distillation for complex reasoning. arXiv preprint arXiv:2605.08741, 2026b. URL https://arxiv.org/abs/2605.08741.

Chujie Zheng, Shixuan Liu, Mingze Li, et al. Group sequence policy optimization. arXiv preprint arXiv:2507.18071, 2025. URL https://arxiv.org/abs/2507.18071.

Shuyan Zhou, Frank F. Xu, Hao Zhu, et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

Yuhang Zhou, Kai Zheng, Haoling Li, Dengyun Peng, Can Xu, and Jingjing Chen. TurnOPD: Making on-policy distillation turn-aware for efficient long-horizon agent training. arXiv preprint arXiv:2607.05804, 2026. URL https://arxiv.org/abs/2607.05804.

A Scope and Limitations

Harness-specific adaptation. HPD trains policies for a specified runtime contract. Applying the method to another harness requires reconstructing its teacher and training a separate adapter, as in Figure 4. The experiments evaluate this per-harness adaptation procedure. Reference replay supplies successful reference prefixes as the training distribution.

Verifier coverage and privileged information. Teacher construction requires exact agreement on the designated harness-policy and tool-call portions of the reference. The TRAJECT-Bench evaluation verifier checks tool names, required keys, control flow, and termination using fixed replay queues for tool outputs. Required-key matching defines its argument-level criterion. The control document combines pre-action observations with reference-ledger and action-witness labels. These labels supply privileged supervision during teacher construction and training; deployment acts directly from the native context. The component ablations evaluate the document views as complete supervision units.

Online extension. Reference replay primarily covers successful recorded prefixes. Extending training to states reached after deployment errors would require verified corrections for those states.

B Replay Construction Details

Figure 6 summarizes the workflow from verified replay through joint training to closed-loop agent deployment.

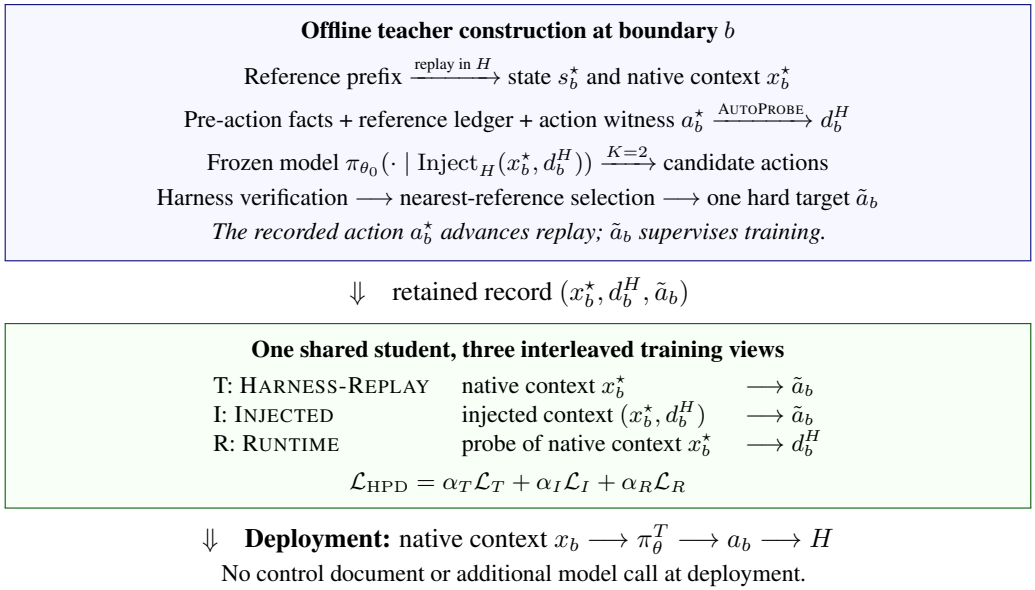


Figure 6: Overview of Harness Policy Distillation. Reference trajectories are replayed through the target harness; at each retained model-call boundary the runtime state is compiled into a control document, the frozen base model is queried with that document, and the verified candidate nearest the recorded trajectory becomes the hard action target. One shared student is trained on native-runtime action prediction, document-conditioned action prediction, and document reconstruction.

B.1 Assumptions and harness adaptation

Replay requires five conditions: each recorded action must be executable under the target harness; the pre-action runtime prefix must be available at every model-call boundary; observed fields must be computed only from that prefix; privileged control annotations must declare their replay provenance; and the successor transition must be verifiable. The recorded action is one executable witness that reconstructs the reference path. The selected verified model candidate supplies the action-training

target. At each boundary, AUTOPROBE compiles the control document, the frozen base model is queried with the document-injected context, and candidate actions are checked by the harness verifier. The main replay state is then advanced with the recorded action, while recorded tool observations are consumed through a per-tool queue stored in runtime state. Unconsumed future observations remain an execution fixture and are not exposed to the model-visible prefix. The construction stores the selected verified hard target rather than teacher logits.

Adapting HPD to a new harness requires definitions of the deployment serializer Ω_H, transition law T_H, decision boundaries, control-state adapter Φ_H, Inject and Probe prompt constructors, and transition and obligation verifiers. The observation adapter may inspect only messages, prior actions, tool results, scheduler events, permissions, and exit state available at the current boundary. The subsequent replay compiler may additionally use the reference stage and current action witness to label ready calls and the continue/final condition. Pi represents lifecycle phase, completed and pending obligations, ready calls, and the continue/final condition. Compiler additionally represents DAG readiness, dispatch, and joins; OpenCode includes permission gates and post-tool continuation. Harness-specific adapters translate these lifecycle structures into control documents; the same replay, verification, and three-view objective then apply to each adapter.

B.2 AUTOPROBE BOUNDARY AND FIELD PROVENANCE

For Pi, the probe is attached to the real provider request inside the agent loop. When the callback fires, the current action has not yet been generated, whereas all prior assistant actions, tool results, errors, and user steering have already been inserted and the previous turn preparation has completed. AUTOPROBE snapshots this state before the privileged model call. The probe location therefore coincides with an actual model decision boundary rather than a token-level approximation; the resulting sampled action is subsequently checked against the harness transition and reference-stage obligation ledger.

The compiler first normalizes harness-visible objects into stable fields. It retains message roles and text, prior tool names and arguments, tool-result content and error status, available tool schemas, and execution modes, while discarding call identifiers, timestamps, usage metadata, streaming events, and runtime function objects. From this pre-action snapshot it deterministically computes observational fields such as lifecycle position, issued calls, and successful or failed tools. A second replay-ledger stage uses the reference stage to label completed and pending recorded requirements. Witness-derived fields use the current recorded action to label the ready frontier and the next continue/final decision. Each field class consequently has a distinct supervisory role: observed fields describe the pre-action runtime, replay-ledger fields identify progress along the reference, and witness-derived fields annotate the next recorded control decision. The complete document is a privileged target. RUNTIME learns its conditional prediction from the native context, while INJECTED receives its explicit action hints. This provenance connects the source of each label to its use in training: the reference supplies the supervised control labels, the injected view learns the associated action, and the native and runtime views learn their conditionals from deployment history.

The provenance distinction used throughout the paper is summarized as follows:

Field class	Source	Prefix-causal
Observed	pre-action harness snapshot	yes
Replay ledger	reference stage structure	no
Witness-derived	current recorded action	no

B.3 TEACHER SAMPLING, VERIFICATION, AND RETRIES

Each sampling attempt produces $K = 2$ complete candidate actions from the frozen base model with the control document injected. The hard gate requires the recorded control decision, tool names, and complete normalized tool arguments to match; tool observations are replayed using the reference parameters. Among candidates passing this gate, selection compares the remaining generated fields with the reference. Writing $r(a)$ for their vector representation, the selection is

$$\mathrm{SelectNearGold}(\mathcal{A}_b^{\mathrm{ver}}; a_b^\star) \in \arg\max_{a \in \mathcal{A}_b^{\mathrm{ver}}} \cos(r(a), r(a_b^\star))\,. \tag{11}$$

The selected verified action is the hard target shared by T and I. Thus the executable tool content remains reference-aligned while similarity selection operates on the other generated fields.

If neither candidate passes verification, construction stops at the failed turn and retries from a prefix extended by one recorded trajectory turn, including its recorded tool content. Each subsequent failure extends the prefix by one more turn before resampling. The model generates each continuation in the harness loop, and only a verified sampled action is retained. The first $K = 2$ attempt accepts 87.90% of raw trajectories, and cumulative coverage reaches 100% by the fifth attempt. Evaluation and deployment execute autonomously. Table 3 summarizes these settings and diagnostics.

Table 3: Teacher sampling settings and recorded construction diagnostics.

Setting or diagnostic	Value
Candidates per sampling attempt	$K = 2$
Attempts to full raw-trajectory coverage	5
Sampling temperature	1
Vocabulary filter	`distribution_top_k=0` (full vocabulary)
First-attempt acceptance rate ($K = 2$)	87.90%
Cumulative coverage by attempt 5	100%
Agreement with the reference trajectory	90.89%

C Experimental Details

C.1 Data construction and evaluation

The experiments use a TRAJECT-Bench partition (He et al., 2026) with 4,703 training tasks, 584 validation tasks, and 583 held-out test tasks, for 5,870 tasks in total. The Pi test set contains 2,140 gold-history action boundaries. Each full task is executed through a multi-turn agent loop with replayed tool outputs. Main end-to-end evaluation uses temperature zero and one rollout per case. The TRAJECT-Bench results are produced by `pi-runtime` using the replay-based metric definitions in Appendix E. The Qwen3-8B Base in the checkpoint and model-size studies uses the same end-to-end scores as Table 1; the portability study reports the Base runs associated with its respective configurations as within-panel references.

The training pool contains 17,599 retained replay boundaries. Each boundary defines one Harness-Replay record, one Injected record, and one Runtime record, for 52,797 boundary–view records. The T and I records share the selected hard target $\tilde{a}_b$, whereas the R record has the point target d_b^H. No additional task trajectory is introduced by the extra views. Contexts follow each backbone's native budget; construction does not impose a fixed truncation that silently removes late turns. Teacher construction follows the sampling, exact-match verification, similarity ranking, and retry procedure in Appendix B.3.

C.2 Training-step convention

Each active view receives 20k batch exposures. In a joint run, one training round cycles through the active views, with one optimizer update per view; for T/I/R, this means three view-specific batch updates, each on 128 sequences. The completed T-only, joint two-view, and joint three-view runs therefore have 20k rounds and log 20k, 40k, and 60k optimizer updates, respectively. A staged two-view run instead applies 20k consecutive updates per view (40k in total) rather than interleaving them in joint rounds.

The optimization diagnostics retain the original optimizer-step axis: 20k, 40k, and 60k updates correspond to approximately 6.7k, 13.3k, and 20k training rounds in the full three-view run. Separate portability and signal-density studies retain their own fixed evaluation protocols.

C.3 Training and comparison methods

The principal experiments use Qwen3-8B (Yang et al., 2025) with LoRA adaptation (Hu et al., 2022). The completed HPD component runs use 20k batch exposures per active view, as defined in

Appendix C.2. The shared training configuration is summarized in Table 4. The portability study trains adapters for Qwen3-8B, Llama3-8B, Gemma4-12B, and Qwen3-MoE on Pi, and separate Qwen3-8B adapters for Pi, ReAct, Compiler, and OpenCode.

Table 4: Training platform and optimization settings shared by HPD and all baseline training runs.

Setting	Value
Accelerators	8× NVIDIA A800-SXM4-80GB (81,920 MiB each)
Host resources	64 CPU cores; 1,030,000 MB RAM
Runtime and precision	CUDA 12.8; BF16 (bfloat16)
Batch size	128
Optimizer	`torch.optim.AdamW(foreach=False)`
Learning rate	constant 2×10^{-6}
Schedule	no scheduler, warmup, or decay
Adam parameters	$\beta = (0.9, 0.999)$; $\epsilon = 10^{-8}$
Weight decay	0.01
Gradient clipping	maximum norm 1.0
Main-run training seed	42

The Qwen3-8B comparison includes the unadapted Base model; standard SFT, which maximizes the likelihood of recorded ground-truth actions under their reference histories; DPO (Rafailov et al., 2023); matched-data GiGPO and GRPO-RLVR adaptations (Feng et al., 2025; Shao et al., 2024); GSPO (Zheng et al., 2025); GDPO (Liu et al., 2026); a Pi adaptation of training with harnesses (OPHSD) (Zhao et al., 2026b); OPSD, using privileged-context self-distillation (Zhao et al., 2026a); classic OPD (Agarwal et al., 2024); and TurnOPD (Zhou et al., 2026). HARNESS-REPLAY is HPD's native-action view. HPD-T retains the full HPD teacher-construction pipeline and is evaluated in Tables 1, 2, and 5. All baseline training runs share the HPD configuration in Table 4; their objectives, preference construction, rollout collection, and use of privileged information differ. Rollout costs and total compute remain method-dependent.

Supervision sources. Table 1 identifies the primary training signal for each method. SFT uses recorded ground-truth actions, and DPO uses preference labels (Rafailov et al., 2023). All RL variants are grouped under *Task reward*; GRPO-RLVR implements this supervision through rule-based verification. OPHSD uses a harness-augmented model teacher (Zhao et al., 2026b); OPSD, OPD, and TurnOPD use model predictions as distillation supervision. For HPD, *Harness runtime* denotes the runtime control documents and harness-verified action targets constructed from reference replay and frozen-model candidates, as defined in Section 3.3.

Distillation baselines. OPSD distills supervision from the base model conditioned on privileged information; OPD and TurnOPD use a privileged 32B teacher. All three use the same `pi-runtime` evaluation pipeline as the other baselines.

C.4 API AND CROSS-BENCHMARK EVALUATION SETTINGS

The API comparison uses the same Pi agent, test tasks, evaluation settings, and metric definitions as Table 1, replacing only the model backend with the endpoint identified in Figure 5. The HPD checkpoint is unchanged. BFCL evaluation instead replaces the test tasks and retains each model's TRAJECT-Bench checkpoint without retraining, as well as the same Pi agent and model-side evaluation settings. Task and tool execution follow BFCL v4's default settings. Appendix C.6 reports this cross-benchmark transfer.

C.5 SEED-STUDY RESULTS

Table 5 reports five additional runs with seeds 715–719. These runs are separate from the main-table run with seed 42. Cases interrupted by device crashes are excluded from the affected run's evaluation, so effective task counts can differ across runs. Each run is scored on its retained cases, and the table reports the mean and standard deviation of the five run-level scores. Base is evaluated under the same five seed settings; its variation reflects differences in inference outcomes across those evaluations, without model training. Full HPD reaches 22.14 mean TS, improving by 19.46 points over Base. Relative to HPD with the T view alone, joint T/I/R training adds 3.94 points in TS, 1.29 in F1, 4.37 in Step-SR, and 5.44 in RC. Both HPD variants use the same teacher-construction procedure; this comparison measures the additional benefit of jointly supervising control documents and actions.

Table 5: Supplementary seed-study results from `pi-runtime`. Values are percentages, reported as mean ± standard deviation over five seeds (715–719), excluding the main-run seed 42. Run-level scores exclude cases interrupted by device crashes. HPD-T retains the full teacher-construction pipeline and trains only the native-action view.

Method	TS	F1	Step-SR	RC
Base	2.68 ± 0.23	26.81 ± 0.11	26.54 ± 0.13	19.62 ± 0.15
HPD-T	18.20 ± 1.67	39.50 ± 0.66	56.21 ± 3.43	42.82 ± 0.81
Full HPD	$\mathbf{22.14 \pm 1.18}$	$\mathbf{40.79 \pm 0.39}$	$\mathbf{60.58 \pm 3.54}$	$\mathbf{48.26 \pm 0.88}$

Full HPD improves all four mean metrics over its T-only ablation. Its TS and F1 standard deviations are 1.18 and 0.39, respectively, compared with 1.67 and 0.66 for T only. The five-run results reinforce the component comparison: harness-verified target construction supports native-action learning, and joint control-document supervision improves completion, reference-prefix execution, and compliance further.

For paired significance testing, we use a two-sided exact sign-flip test on source-cluster differences across the five seed runs. We apply Holm correction within each comparison to TS, F1, Step-SR, and RC. The adjusted TS p-values for Full HPD versus HPD-T and OPD are 0.00343 and 0.00338, respectively.

C.6 CROSS-BENCHMARK TRANSFER WITHOUT RETRAINING

We evaluate Base, SFT, DPO, GRPO, OPD, OPSD, HPD-T, and HPD on 92 BFCL v4 cases using the same checkpoints as in the TRAJECT-Bench comparison, without further training on BFCL. The Pi agent and model-side evaluation settings remain unchanged; only the test tasks are replaced. Task and tool execution follow BFCL v4's default settings. BFCL evaluates function calling and stateful agentic behavior (Patil et al., 2025). Figure 7 reports TS, final-answer F1, and Step-SR, using the same metric definitions as the main comparison. This BFCL evaluation reports these three metrics.

Full HPD has the highest TS and Step-SR (21.74 and 48.12), whereas HPD-T has the highest F1 (26.12). HPD-T matches Base's 13.04 TS but exceeds its F1 and Step-SR by 6.66 and 4.13 points. Relative to HPD-T, Full HPD gains 8.70 TS and 1.96 Step-SR points while losing 2.83 F1 points. T-only replay thus transfers answer overlap and prefix execution without improving completed-task coverage on this collection; the additional I/R views are associated with the completion gain. OPD and DPO tie for the highest external-baseline TS at 17.39. OPD has the strongest external-baseline F1 and Step-SR (21.50 and 45.87); Full HPD exceeds it by 4.35, 1.79, and 2.25 points, respectively. OPSD reaches 8.70 TS, 16.89 F1, and 44.06 Step-SR, improving reference-prefix execution over Base without improving task completion. DPO improves TS over Base but lowers F1 and Step-SR; SFT and GRPO fall below Base on all three metrics. These differences suggest that verified replay alone can transfer partial execution behavior, while I/R better connects those actions to the runtime control state required for completion on new tasks.

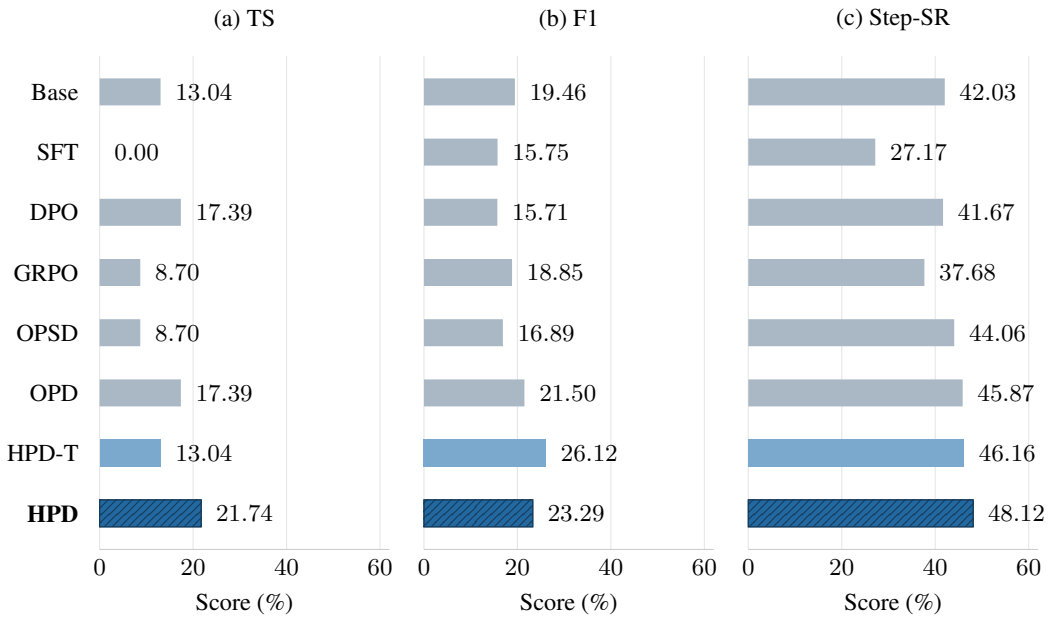


Figure 7: BFCL v4 transfer on 92 cases without retraining. Panels show (a) TS, (b) final-answer F1, and (c) Step-SR for the eight methods. All axes share the same zero-based percentage scale. HPD-T and Full HPD are blue; Full HPD is hatched. Numerical labels give the scores on this subset.

D Optimization Diagnostics

D.1 Checkpoint dynamics

Figure 8 aligns optimization loss with end-to-end performance at 20k, 40k, and 60k optimizer steps in the full three-view run. The 200-cycle mean loss decreases from 0.188 to 0.181 and 0.162. TS rises from 3.09 to 16.12, 19.55, and 22.98, while F1 rises from 26.82 to 39.90, 39.79, and 41.22. Continued training improves executable cross-turn completion after most answer-content gains have appeared.

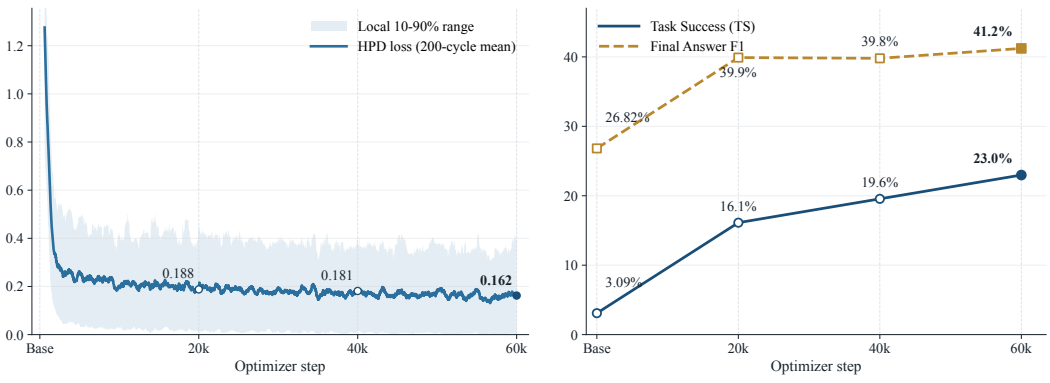


Figure 8: Full HPD optimization over 60k optimizer steps (20k training rounds). Left: 200-cycle mean loss and local 10–90% range. Right: task success and final-answer F1. Base scores are TS 3.09% and F1 26.82%; trained-checkpoint labels are rounded to one decimal.

Table 6: Intermediate and completed full-HPD checkpoints evaluated by `pi-runtime`, indexed by optimizer steps. The 60k endpoint completes 20k training rounds. Scores are percentages.

Optimizer steps	TS	F1	Action-Acc	Step-SR	RC	Exact
40k (intermediate)	19.55	39.79	63.74	58.27	43.40	15.61
60k (completed)	22.98	41.22	64.95	63.79	49.23	18.70

D.2 Loss decomposition

Because each optimizer update uses one view, a naive moving average mixes objective composition with optimization progress. Aggregating each complete T/I/R cycle yields the trend in Figure 8. At the 60k-optimizer-step endpoint, the view losses are 0.2415 for Harness-Replay, 0.2283 for Injected, and 0.0170 for Runtime. The losses summarize different prediction tasks with completion-length normalization. In particular, the document loss measures token prediction across observed, ledger, and witness-derived fields. Reporting the losses separately tracks optimization progress for each supervision path; rollout metrics measure the resulting policy's execution behavior.

D.3 Signal-density search

Figure 9 and Table 7 report a separate Qwen3-8B diagnostic at 20k optimizer steps per configuration. Harness-Replay is fixed at 1.0 while either Injected or Runtime is varied over 0.2, 0.6, 1.4, and 1.8; the other auxiliary view stays at 1.0. These weights set the sampling frequency of each view. At the fixed update budget, changing one frequency reallocates updates among the three supervision paths. The default 1:1:1 cycle anchors both sweeps. Injected at 0.2 gives the highest RC (43.57) and Step-SR (56.40), gains of 2.92 and 3.64 percentage points over the default. Runtime at 0.6 gives the highest TS (18.18), Exact (14.24), and F1 (40.05), improving by 2.06, 1.03, and 0.15 points, respectively. The preferred frequency depends on the target metric: Injected at 0.2 favors prefix execution and compliance, whereas Runtime at 0.6 favors completed tasks in this sweep. Each axis is searched separately with one seed, giving a coordinate-wise comparison around the default cycle.

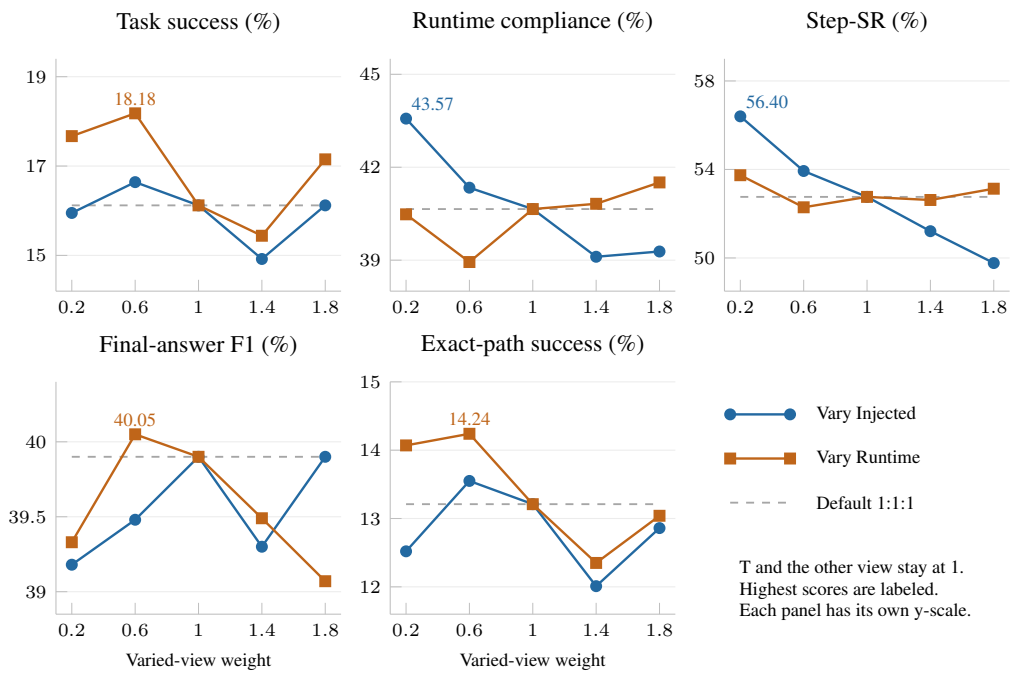


Figure 9: Single-axis sampling-weight search at 20k optimizer steps. Blue circles vary Injected; orange squares vary Runtime. Dashed lines mark the shared 1:1:1 default. Each panel shows one metric with its own y-scale and labels its maximum. Lines connect evaluated settings.

Table 7: Complete single-axis sampling-weight search. Ratios give the realized T:I:R sampling cycle; all scores are percentages. Bold marks the highest score in each metric.

Varied view	T:I:R	TS	F1	Step-SR	RC	Exact
Default	1:1:1	16.12	39.90	52.76	40.65	13.21
Injected (0.2)	5:1:5	15.95	39.18	**56.40**	**43.57**	12.52
Injected (0.6)	5:3:5	16.64	39.48	53.93	41.34	13.55
Injected (1.4)	5:7:5	14.92	39.30	51.21	39.11	12.01
Injected (1.8)	5:9:5	16.12	39.90	49.77	39.28	12.86
Runtime (0.2)	5:5:1	17.67	39.33	53.74	40.48	14.07
Runtime (0.6)	5:5:3	**18.18**	**40.05**	52.29	38.94	**14.24**
Runtime (1.4)	5:5:7	15.44	39.49	52.62	40.82	12.35
Runtime (1.8)	5:5:9	17.15	39.07	53.13	41.51	13.04

D.4 Model-size scaling study

Table 8 evaluates Base and HPD across seven Qwen3 checkpoints. HPD improves both TS and F1 at every size, with TS gains ranging from 4.46 points at 0.6B to 14.41 at 14B; the 32B gain is 13.90 points. Step-SR and RC also improve at every size. The 14B and 32B dense checkpoints exceed the 30B-A3B MoE checkpoint on TS. All rows use the same 583 cases and one seed.

Each Base evaluation contains 583 unique cases and 2,140 reference steps, with no worker failures. TS uses the historical task-success definition; F1, RC, and Exact use the runtime evaluation summaries. Action-Acc comes from a separate native gold-history boundary evaluation. The Qwen3-8B HPD row corresponds to the 20k-optimizer-step checkpoint, rather than the completed 60k checkpoint in Table 1.

Table 8: Qwen3 model-size results. Action-Acc uses the native gold-history boundary protocol separately from end-to-end evaluation. Exact retains the historical strict trajectory-success definition, not answer-string exact match. All values are percentages.

Model	Variant	TS	F1	Action-Acc	Step-SR	RC	Exact
Qwen3-0.6B	Base	0.00	24.78	52.06	3.64	0.00	0.00
	HPD	4.46	25.70	54.07	24.20	16.50	2.40
Qwen3-1.7B	Base	0.00	27.00	49.90	4.53	0.00	0.00
	HPD	6.52	30.50	59.00	33.60	20.80	3.30
Qwen3-4B	Base	1.03	20.81	53.40	9.81	7.38	0.86
	HPD	15.09	38.50	59.40	48.60	38.90	11.50
Qwen3-8B	Base	3.09	26.82	53.40	26.45	19.90	2.57
	HPD	16.12	39.90	60.80	52.80	40.70	13.21
Qwen3-14B	Base	3.43	27.96	52.90	14.77	15.61	2.74
	HPD	17.84	40.00	60.00	57.40	44.90	14.92
Qwen3-30B-A3B	Base	4.12	28.86	52.90	14.44	17.67	2.74
	HPD	16.81	38.90	58.80	53.60	40.10	13.21
Qwen3-32B	Base	6.00	30.42	56.82	19.63	25.73	4.12
	HPD	19.90	40.29	62.15	64.49	49.23	16.81

E Metric Definitions

Unless noted otherwise, reported values are percentages. Final-answer F1 is token-level F1 between the emitted final response and the reference answer.

Gold-history action accuracy. At each of the 2,140 Pi provider boundaries, the evaluator supplies the recorded gold history as context and compares the sequence likelihood of the reference next action with that of a constructed alternative. Action-Acc is the fraction of boundaries where the

reference action ranks higher. The two candidates are scored but not executed; each boundary is evaluated independently, so model-generated rollout histories do not enter this metric. The rollout metrics below evaluate the policy through execution.

Reference-prefix step success. Step-SR counts correct reference steps until first divergence:

$$\text{Step-SR} = \frac{\sum_i \text{correct prefix steps in rollout } i}{\sum_i \text{reference steps in task } i}. \tag{12}$$

Later actions receive no reference-path credit after the first divergence by definition, even if the rollout later recovers or reaches an equivalent runtime state.

Runtime compliance, Exact, and task success. RC is the fraction of tasks satisfying the harness execution contract: required tool sequence and parameter keys, exactly one final response, native exit, and a healthy worker. Exact additionally requires reproduction of the reference control path and final-answer F1 of at least 0.5. For task i, let O_i indicate eventual obligation completion, F_i final-answer F1, C_i exactly one final, E_i native exit, and W_i healthy worker execution:

$$\text{TS}_i = \mathbf{1}[O_i]\mathbf{1}[F_i \geq 0.5]\mathbf{1}[C_i]\mathbf{1}[E_i]\mathbf{1}[W_i]. \tag{13}$$

TS is the mean of this indicator. A recovered success can have Exact$= 0$ because retries or a valid action ordering different from the reference path are allowed. The TRAJECT-Bench evaluation contract uses required parameter keys as its argument-level criterion and replayed observations as tool outputs. RC, Exact, and TS therefore describe execution within this specified runtime.

E.1 Token-distribution diagnostic

At boundary b, let $c_b^\star$ be the correct initial control token and $u^\star_{b,1:m_b}$ the reference continuation after it, truncated to $m_b \leq 63$ tokens. For $m_b > 0$, define

$$\begin{aligned} p_{b,j}(v) &= \pi_\theta(v \mid x_b^\star, c_b^\star, u^\star_{b,<j}), \\ \bar{h}_b &= -\frac{1}{m_b}\sum_{j=1}^{m_b}\sum_{v\in\mathcal{V}} p_{b,j}(v)\log_2 p_{b,j}(v), \\ g_b &= \exp\left(\frac{1}{m_b}\sum_{j=1}^{m_b}\log p_{b,j}(u^\star_{b,j})\right), \end{aligned} \tag{14}$$

where $\mathcal{V}$ is the full token vocabulary. The three panels of Figure 3 plot $\pi_\theta(c_b^\star \mid x_b^\star)$, $\log_{10}\bar{h}_b$, and g_b, respectively. Thus panel (b) takes the logarithm *after* averaging position-wise entropy, while panel (c) exponentiates the mean reference-token log probability. Neither continuation statistic uses model-generated prefixes.

E.2 Interpreting F1 and task completion

For task-averaged final-answer F1, let S_i be the binary task-success indicator and $F_i \in [0,1]$ the final-answer F1 for task i. Writing $p_{\text{TS}} = \mathbb{E}[S_i]$, the mean decomposes as

$$\mathbb{E}[F_i] = p_{\text{TS}}\,\mathbb{E}[F_i \mid S_i = 1] + (1 - p_{\text{TS}})\,\mathbb{E}[F_i \mid S_i = 0]. \tag{15}$$

Thus aggregate F1 can change through the fraction of successful tasks, the mean F1 within either group, or both. The tables provide the aggregate and success rate, which describe answer overlap and completed-task coverage as separate outcomes.

HPD and OPD have nearly equal aggregate F1 (41.22 versus 41.04), while HPD has substantially higher TS (22.98 versus 15.61) and Exact (18.70 versus 14.07). OPSD similarly raises TS over Base (13.55 versus 3.09) without raising F1 (26.27 versus 26.82). Together with HPD's T-only and joint-view ablations, these patterns identify HPD's completion advantage at comparable aggregate answer overlap. The distinction follows directly from the metrics: F1 averages answer overlap over all tasks, whereas TS counts tasks that jointly satisfy the execution conditions and answer threshold. Similar mean F1 can therefore accompany different successful-completion coverage.

Equation 15 also specifies a task-level analysis for attributing F1 changes: measure answer overlap on tasks completed by both methods, newly completed tasks, previously completed tasks that become unsuccessful, and tasks completed by neither method. Grouping instead by execution-only conditions, such as obligation completion and native termination, separates execution coverage from the answer threshold used in TS. These two decompositions distinguish the coverage of successful execution from answer quality within each group. Runtime alignment and reference-distribution fitting can contribute jointly; paired, execution-conditioned analyses separate their completion-coverage and within-group answer-quality outcomes.

F Distillation Derivation and Implementation

F.1 Replay-teacher distributions

Offline construction fixes a document and selected action at each retained boundary, defining the point-mass teacher. Here δ_z assigns probability one to the sequence z and zero elsewhere; d and a are document and action variables, respectively:

$$\pi_b^{\mathrm{rep}}(d, a \mid x_b^\star) = \delta_{d_b^H}(d)\ \delta_{\tilde{a}_b}(a). \tag{16}$$

Write $\pi^{\mathrm{rep}}(d, a \mid x)$ for the empirical conditional distribution of document–action pairs given context x. Its document marginal π_R^{rep}, document-conditioned action distribution π_I^{rep}, and native action marginal π_T^{rep} satisfy

$$\pi^{\mathrm{rep}}(d, a \mid x) = \pi_R^{\mathrm{rep}}(d \mid x)\pi_I^{\mathrm{rep}}(a \mid x, d), \tag{17}$$

$$\pi_T^{\mathrm{rep}}(a \mid x) = \sum_d \pi^{\mathrm{rep}}(d, a \mid x). \tag{18}$$

The document includes runtime facts and privileged replay labels. Each record has a point target; records sharing a serialized context induce an empirical mixture when conditioned on that context. This distinction connects stored supervision to the conditional teacher objectives.

F.2 Conditional KL decomposition

All KL divergences use natural logarithms. Expectations below are taken under the empirical teacher conditionals defined above.

Runtime and Injected define the factorized student $\widetilde{\pi}_\theta(d, a \mid x) = \pi_\theta^R(d \mid x)\pi_\theta^I(a \mid x, d)$. Suppressing boundary and harness indices, the KL chain rule gives

$$\begin{aligned} &D_{\mathrm{KL}}(\pi^{\mathrm{rep}}(d, a \mid x) \,\|\, \widetilde{\pi}_\theta(d, a \mid x)) \\ &\quad = D_{\mathrm{KL}}\big(\pi_R^{\mathrm{rep}}(\cdot \mid x) \,\|\, \pi_\theta^R(\cdot \mid x)\big) \\ &\qquad + \mathbb{E}_{d\sim\pi_R^{\mathrm{rep}}(\cdot|x)} D_{\mathrm{KL}}\big(\pi_I^{\mathrm{rep}}(\cdot \mid x, d) \,\|\, \pi_\theta^I(\cdot \mid x, d)\big)\,. \end{aligned} \tag{19}$$

The chain rule motivates the R and I losses; T additionally matches the teacher's native-runtime action marginal. The three losses in Equation 9 are

$$\begin{aligned} \mathcal{L}_R &= \mathbb{E}_{x\sim\mu_H^{\mathrm{rep}}} D_{\mathrm{KL}}\big(\pi_R^{\mathrm{rep}}(\cdot \mid x) \,\|\, \pi_\theta^R(\cdot \mid x)\big)\,, \\ \mathcal{L}_I &= \mathbb{E}_{x\sim\mu_H^{\mathrm{rep}}} \mathbb{E}_{d\sim\pi_R^{\mathrm{rep}}(\cdot|x)} D_{\mathrm{KL}}\big(\pi_I^{\mathrm{rep}}(\cdot \mid x, d) \,\|\, \pi_\theta^I(\cdot \mid x, d)\big)\,, \\ \mathcal{L}_T &= \mathbb{E}_{x\sim\mu_H^{\mathrm{rep}}} D_{\mathrm{KL}}\big(\pi_T^{\mathrm{rep}}(\cdot \mid x) \,\|\, \pi_\theta^T(\cdot \mid x)\big)\,, \end{aligned} \tag{20}$$

where π_T^{rep} is the deployment marginal of Equation 18.

F.3 Stored targets and the implemented loss

After sampling, verification, and near-gold selection, let c_b^v denote the stored prompt and p_b^v the target-sequence distribution for boundary b and view $v \in \{T, I, R\}$. The three implementation pairs are

$$\begin{aligned} (c_b^T, p_b^T) &= (x_b^\star, \delta_{\tilde{a}_b})\,, \\ (c_b^I, p_b^I) &= \big(\mathrm{Inject}_H(x_b^\star, d_b^H), \delta_{\tilde{a}_b}\big)\,, \\ (c_b^R, p_b^R) &= \Big(\mathrm{Probe}_H(x_b^\star), \delta_{d_b^H}\Big)\,. \end{aligned} \tag{21}$$

Thus the two action views share the selected verifier-approved hard target, while the document view predicts the control document stored for the same boundary. Student training consumes these stored targets and requires no teacher forward pass. Rejected and unselected candidates do not enter the three supervised targets.

Per-record NLL and conditional teacher KL. Equation 10 establishes the point-target identity for each complete stored sequence. To connect these per-record losses to the conditional teacher, average over the empirical distribution of prompt–target pairs.

For any frozen target distribution p and context c, with $\mathcal{H}(p)$ denoting Shannon entropy using natural logarithms,

$$\mathbb{E}_{y\sim p}\left[-\log \pi_\theta(y \mid c)\right] = D_{\mathrm{KL}}(p \,\|\, \pi_\theta(\cdot \mid c)) + \mathcal{H}(p), \tag{22}$$

where $\mathcal{H}(p)$ is independent of θ (Kim & Rush, 2016). Let $\widehat{P}_v(c, y)$ be the empirical distribution of stored pairs for view v, with context marginal $\widehat{P}_v(c) = \sum_y \widehat{P}_v(c, y)$ and conditional teacher $q_v(y \mid c) = \widehat{P}_v(y \mid c)$. Conditioning on the prompt and applying Equation 22 gives

$$\begin{aligned}\mathbb{E}_{(c,y)\sim\widehat{P}_v}[-\log \pi_\theta(y \mid c)] = \mathbb{E}_{c\sim\widehat{P}_v}\left[D_{\mathrm{KL}}(q_v(\cdot \mid c)\|\pi_\theta(\cdot \mid c))\right] \\ + \mathbb{E}_{c\sim\widehat{P}_v}\mathcal{H}(q_v(\cdot \mid c)).\end{aligned} \tag{23}$$

For T, I, and R, q_v is the corresponding conditional teacher under Equation 21. Thus per-record NLL averaging recovers $\mathcal{L}_T$, $\mathcal{L}_I$, and $\mathcal{L}_R$ up to constant entropies. Individual targets have zero entropy; pooling records with a shared prompt produces the conditional mixture in Equation 23.

Normalization and implementation. For a target y of $|y|$ tokens, y_j is token j and $y_{<j}$ its preceding prefix. The trainer uses the completion-length-normalized loss

$$\ell_\theta(y, c) = -\frac{1}{|y|}\sum_{j=1}^{|y|}\log \pi_\theta(y_j \mid c, y_{<j}), \tag{24}$$

which normalizes short documents, final responses, and longer tool-call waves by their completion lengths. For the stored point targets, this is the sequence-level KL in Equation 10 weighted by $1/|y|$.

The implemented objective samples retained records uniformly from $\mathcal{D}_H^{\mathrm{rep}}$ and selects view v with probability α_v:

$$\mathcal{L}_{\mathrm{HPD}}^{\mathrm{impl}} = \mathbb{E}_{(x_b^\star, d_b^H, \tilde{a}_b)\sim\mathcal{D}_H^{\mathrm{rep}}}\mathbb{E}_{v\sim\mathrm{Cat}(\alpha_T,\alpha_I,\alpha_R)}\mathbb{E}_{y\sim p_b^v}\left[\ell_\theta(y, c_b^v)\right]. \tag{25}$$

Each inner expectation evaluates the stored target: $\tilde{a}_b$ for T/I and d_b^H for R. A deterministic cycle realizes the long-run view weights through sequential AdamW updates, each using 128 sequences from one view's seeded permutation. One T/I/R round is one training step, giving default weights $\alpha_T = \alpha_I = \alpha_R = 1/3$ and 60k optimizer updates over 20k training steps. All views update the same parameters; deployment uses only $a_b \sim \pi_\theta^T(\cdot \mid x_b)$ under the native harness contract.