TRAINING CONTRACTS: LEARNING VALIDITY BOUNDARIES FOR REUSING TRAINING INTERVENTIONS IN EMBODIED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied reinforcement learning accumulates experience from training interventions, which may be transferred to future learning processes. However, the same intervention can have different effects under different learning contexts, making it difficult to determine when an intervention remains valid for reuse. We introduce Training Contracts, which learn an explicit validity boundary for each training intervention by comparing training processes with and without each intervention in the source domain. Each contract records supported, contradicted, and unresolved regions, and validates the boundary on independent source checkpoints. For target tasks, the trainer uses the validity boundary to decide whether to reuse an intervention, perform a probe, or abstain. All target policies are trained from scratch, and only external intervention knowledge is transferred. Across twelve target configurations from six task families, Training Contracts improve average performance from 0.59 to 0.64 (8.5% relative gain) over the strongest conditional-effect prediction baseline. Beyond the primary benchmark, including Meta-World and real-robot experiments, Training Contracts improve success rates by 14.5% and 15.1% over the average baseline performance, respectively.

1 INTRODUCTION

Embodied reinforcement learning can accumulate reusable experience through training interventions that alter how an agent learns, including reward edits, exploration settings, optimization changes, and curricula. However, the usefulness of an intervention is often conditional: an intervention that improves one learning regime may become ineffective or harmful in another. The central challenge is therefore not only to remember which intervention worked, but to identify the validity boundary that determines where previous evidence supports reuse, where it contradicts reuse, and where further evidence is needed.

This paper focuses on transferring training experience rather than policy parameters. An outer trainer chooses interventions while an inner reinforcement-learning algorithm updates the policy. Target policies are trained from scratch, and only records of previous training decisions and their outcomes are transferred.

Consider a robot learning to manipulate an object. Reducing an auxiliary approach reward can disrupt learning before the robot can reliably reach the object, yet help once reaching is reliable and completion remains difficult. A remembered recipe—"reduce the approach reward"—loses the conditions under which that recipe was effective. A conditional-effect predictor can estimate whether an intervention is likely to help at a given context, but reuse still requires identifying the boundary separating contexts where the evidence applies from those where it does not.

We introduce *Training Contracts* to represent this validity boundary. A contract maintains explicit regions where an intervention is supported, contradicted, or unresolved, and validates the boundary on independent source checkpoints before direct reuse. During target learning, the trainer uses the confirmed contract to reuse an intervention, perform a bounded probe, or abstain. Figure 1 illustrates the source-to-target lifecycle. Section 3 formalizes the transfer setting, and Section 4 presents the overall method.

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

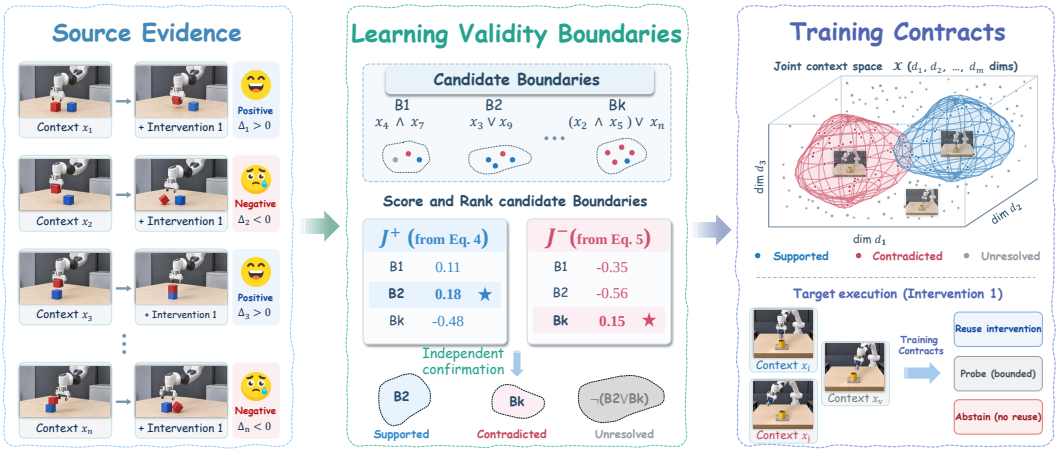


Figure 1: **Overview of Training Contracts.** Using Intervention 1 as an example, source training produces paired comparisons between applying and not applying an intervention, providing positive, negative, and inconclusive evidence across contexts. Training Contracts transform this evidence into explicit validity boundaries that separate supported, contradicted, and unresolved regions in the context space. Candidate boundaries are generated from a finite set of admissible combinations, then scored and ranked before being confirmed on independent source checkpoints. The confirmed contract guides target-time decisions to reuse, probe, or abstain.

A conditional-effect predictor is a natural comparison because it receives the same source evidence and can also model uncertainty. We compare contracts with such predictors and experience-memory controls under matched observations and resource budgets. Across twelve target configurations from six task families, Training Contracts improve target-task performance over strong baselines while retaining the advantage when methods acquire source evidence independently. Additional analyses examine the role of validity regions, coverage, and probing.

Our contributions are threefold:

- We formulate trainer-experience transfer as a validity-boundary problem over finite intervention evidence, rather than policy transfer or context-free recipe reuse.
- We introduce Training Contracts, which learn executable validity boundaries from paired evidence, require independent confirmation for direct reuse, and retain counterevidence.
- We compare contracts with strong conditional-effect and experience-reuse controls under shared evidence and independent source acquisition, with diagnostics for validity regions, coverage, probing, and downstream prediction.

2 Related Work

Learning to train and transfer. Population-based training, meta-gradients, learned optimizers, curricula, and environment design adapt how a learner is trained (Jaderberg et al., 2017; Xu et al., 2018; 2020; Andrychowicz et al., 2016; Franceschi et al., 2018; Pedregosa, 2016; Matiisen et al., 2020; Portelas et al., 2020; Graves et al., 2017; Florensa et al., 2018; Dennis et al., 2020; Jiang et al., 2021). Meta-learning and meta-RL instead transfer initializations or adaptation procedures (Finn et al., 2017; Nichol et al., 2018; Duan et al., 2016; Wang et al., 2016; Rakelly et al., 2019; Zintgraf et al., 2020). Our setting transfers only external records of training interventions. LLM-based methods such as Eureka, DrEureka, and Text2Reward generate reward or environment changes (Ma et al., 2024a;b; Xie et al., 2024). The goal of this paper is to determine when prior intervention evidence remains valid for reuse.

Conditional effects, selective decisions, and memory. Conditional-effect methods estimate heterogeneous intervention effects (Johansson et al., 2016; Shalit et al., 2017; Athey & Imbens, 2016;

Wager & Athey, 2018; Athey et al., 2019; Künzel et al., 2019). Uncertainty and selective prediction support decisions with limited evidence (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Ovadia et al., 2019; El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017; Romano et al., 2020; Angelopoulos & Bates, 2021). SafeOpt and conformal risk control address related uncertainty-aware decision problems (Sui et al., 2015; Angelopoulos et al., 2024). Experience-memory systems organize or evolve reusable feedback (Shinn et al., 2023; Xu et al., 2025; Yang et al., 2026; Zhang et al., 2026b;a; 2025). Training Contracts instead keep explicit regions where source evidence supports reuse, contradicts it, or remains unresolved, together with the underlying counterevidence. We compare against both predictor and memory controls and test observed context changes and a hidden shift in state-based manipulation (Packer et al., 2018; Kirk et al., 2023; Cobbe et al., 2019; 2020; Yu et al., 2020; Fujimoto et al., 2019; Kumar et al., 2020; Gu et al., 2023; Yu et al., 2025; Schulman et al., 2017; Haarnoja et al., 2018).

3 From Training Experience to Reuse Decisions

Training interventions have context-dependent effects. Source training provides evidence that an intervention u can change the learning process under different predecision contexts. The same intervention may improve some learning trajectories while having limited or negative effects in others. Therefore, transferring training experience requires identifying when previous evidence remains applicable rather than simply remembering whether an intervention succeeded before.

For a predecision context x and intervention u, let $\Delta(x, u)$ denote the expected utility difference between applying u and continuing the declared comparator under the same training horizon:

$$\Delta(x, u) = \mathbb{E}[U(T \oplus u) - U(T) \mid x]. \tag{1}$$

Here x contains learner diagnostics and training-state features available before the decision, rather than an environment or MDP state. Utility is measured as empirical task-success frequency under the fixed public goal predicate. The conditional effect $\Delta(x, u)$ motivates a boundary representation because the same intervention can have different effects across regions of the learning context.

Validity boundaries for reuse decisions. Given finite and uneven intervention evidence, the transfer problem is to identify where previous evidence supports reuse, where it argues against reuse, and where it is insufficient for a decision. We refer to these cases as *supported*, *contradicted*, and *unresolved* contexts. The goal is therefore not only to estimate intervention effects, but also to determine the boundary between contexts where prior experience can be applied and contexts where additional evidence is needed.

At target time, this formulation leads to a selective decision problem. The trainer can reuse an intervention when its validity is supported by available evidence, gather additional evidence through a bounded probe when needed, or abstain when reuse cannot be justified. Section 4 introduces the Training Contracts that represent and execute these validity boundaries.

Evaluation objective. At interaction count b_j, the trainer commits a checkpoint using only training and development observations. Let $s_j(P, \ell, r)$ denote the held-out success of the checkpoint selected at budget boundary b_j, where P, ℓ, and r index the target task, independent source state, and target training trajectory, respectively. With budget B,

$$C_{\text{anytime}} = \mathbb{E}_{P,\ell,r}\left[\sum_{j=1}^{J} \frac{b_j - b_{j-1}}{B} s_j(P, \ell, r)\right], \quad b_0 = 0. \tag{2}$$

This score summarizes held-out success over checkpoints selected throughout training. It is applied identically to all methods. Held-out scores never guide checkpoint selection, and probes and failed branches consume the same training budget.

4 Learning and Using Training Contracts

A training intervention may help in some learning contexts but fail in others. Training Contracts model this dependency by learning explicit validity boundaries from paired source comparisons. The

Algorithm 1 Learning and Executing Training Contracts

Require: Source training comparisons, intervention set $\mathcal{U}$, admissible predicate grammar Φ, validation split, reuse threshold η
Ensure: Contract set $\mathcal{C}$ and target-time intervention decisions
Contract learning:
for each intervention $u \in \mathcal{U}$ **do**
  Collect paired source comparisons with and without u
  Generate candidate validity boundaries $\phi(x) \in \Phi$
  Select supported $\phi^+(x)$ and contradicted $\phi^-(x)$ regions using the discovery objective
  Validate the selected boundary on independent source checkpoints
  Store validated contract c_u in $\mathcal{C}$
end for
Target-time execution:
for each target decision context x **do**
  Evaluate contract conditions and apply Eq. (8) to select reuse, probe, or abstain
end for

method has three stages: learning boundaries, confirming them on independent source checkpoints, and making target-time decisions through reuse, probing, or abstention. Algorithm 1 summarizes the procedure. An overview of this process is shown in Figure 1.

4.1 From interventions to validity boundaries

A training contract identifies where an intervention is supported, contradicted, or unresolved rather than assigning it a single global effect. Instead of estimating one global intervention value, the contract explicitly represents the context-dependent validity of the intervention. For intervention u, we represent a contract as

$$c_u = (u, \phi_u^+, \phi_u^-, \sigma_u). \tag{3}$$

Here the first component specifies the intervention being transferred, while ϕ_u^+ and ϕ_u^- denote the supported and contradicted validity regions, respectively. They characterize where intervention u is expected to help or harm based on source evidence, while contexts satisfying neither predicate remain unresolved. The state σ_u records whether the contract is active, restricted, or deprecated.

4.2 Learning validity boundaries from paired evidence

The discovery stage learns validity boundaries from paired source comparisons. Each comparison starts from the same source checkpoint and measures the difference between applying and not applying an intervention. The context contains information available before intervention, including training progress, controller information, goal metadata, and action or critic diagnostics. Positive, negative, and inconclusive comparisons provide evidence about different parts of the validity boundary.

We search a finite set of interpretable predicates built from these context variables. The predicate grammar is fixed before evaluation, while candidate thresholds and combinations are generated from source evidence during discovery. Each predicate $\phi(x)$ is constructed from atomic conditions over the context variables. An atom represents a basic condition, such as a threshold comparison $x_j > \theta$ or $x_j \leq \theta$, and multiple atoms can be combined to form a predicate that defines a candidate validity region. A predicate does not directly predict the intervention effect; instead, it selects a region of the context space where the observed intervention evidence is consistent. During discovery, ϕ denotes a candidate predicate being evaluated, while the optimized supported and contradicted boundaries stored in a contract are denoted as ϕ_u^+ and ϕ_u^-, respectively. Thus, ϕ_u^+ and ϕ_u^- are obtained by optimizing over candidate predicates ϕ for intervention u:

$$J^+(\phi, u) = \frac{1}{n(\phi)} \sum_i \phi(x_i)\big[\Delta_i - \gamma(-\Delta_i)_+\big] - \tau \ \mathrm{Atoms}(\phi). \tag{4}$$

where Δ_i denotes the measured paired source effect of the i-th source comparison, computed as the difference in training outcomes between applying and not applying intervention u under the same source context. The normalization term $n(\phi) = \sum_i \phi(x_i)$ denotes the number of source comparisons

covered by candidate predicate ϕ. The predicate value $\phi(x_i) \in \{0, 1\}$ indicates whether the context is included in the candidate region. The first term rewards regions with positive intervention effects while penalizing harmful assignments. The complexity term $\mathrm{Atoms}(\phi)$ favors compact validity boundaries by discouraging predicates with excessive conditions. The contradicted region is obtained using the reversed objective, which searches for regions with consistently negative paired effects.

$$J^-(\phi, u) = \frac{1}{n(\phi)} \sum_i \phi(x_i) \big[-\Delta_i - \gamma(\Delta_i)_+ \big] - \tau \ \mathrm{Atoms}(\phi). \tag{5}$$

This objective favors contexts where applying the intervention consistently decreases performance while penalizing incorrect sign assignments. Details of the predicate grammar and candidate generation are provided in Appendix E. The selected supported and contradicted regions are disjoint, while contexts without sufficient evidence remain unresolved.

4.3 Independent confirmation of learned boundaries

After selecting a candidate boundary, the intervention and predicates are fixed, and the boundary is evaluated on independent source checkpoints. This step filters boundaries that do not generalize beyond the discovery evidence.

Let $s(x)$ denote the confirmation stratum associated with context x. $L_e(s)$ and $U_e(s)$ denote the lower and upper bounds of the intervention effect computed from independent confirmation samples within stratum s. The parameter ϵ specifies the tolerated small negative deviation during confirmation, while η specifies the minimum lower-bound margin required for direct reuse. A supported region is admitted when

$$\min_{s \in \mathcal{C}_e^+} L_e(s) \geq -\epsilon \quad \text{and} \quad \max_{s \in \mathcal{C}_e^+} L_e(s) > \eta, \tag{6}$$

A contradicted region is confirmed if the upper bound of the intervention effect is sufficiently negative:

$$U_e(s) < -\eta, \qquad s \in \mathcal{C}_e^-. \tag{7}$$

Direct reuse requires the matched target context to satisfy the confirmed contract condition.

4.4 Executing contracts by reuse, probe, or abstention

At target time, the trainer evaluates each intervention u under the current target context x and outputs a decision $d_t(u)$ indicating whether to reuse, probe, or abstain. The decision rule prioritizes safety: contradicted regions are rejected first, confirmed supported regions enable direct reuse, and remaining cases fall back to probing or abstention.

$$d_t(u) = \begin{cases} \textsc{Abstain}, & \phi_u^-(x) = 1, \\ \textsc{Reuse}, & \sigma_u = \text{active},\ \phi_u^-(x) = 0,\ \phi_u^+(x) = 1,\ L_u(s(x)) > \eta,\ \text{checks pass}, \\ \textsc{Probe}, & \sigma_u \in \{\text{active}, \text{restricted}\},\ \phi_u^-(x) \neq 1,\ \neg\textsc{Reuse},\ \text{probe feasible}, \\ \textsc{Abstain}, & \text{otherwise}. \end{cases} \tag{8}$$

Reuse requires both a supported validity boundary and sufficient confirmed evidence. When direct reuse is not justified, the trainer performs a bounded probe when feasible; otherwise it abstains.

5 Experiments

5.1 Experimental setup

Tasks and transfer. Source experience comes from PickCube-v1, PokeCube-v1, LiftPegUpright-v1, and PlaceSphere-v1. StackCube-v1, PushCube-v1, and PullCube-v1 are development tasks. The primary target suite contains twelve configurations across six families: stack-and-release, directional contact, insertion/alignment, articulated manipulation, precision placement, and sequential relocation. Every target policy starts from scratch and only frozen trainer memory transfers. We also evaluate the same source-trained state on the official Meta-World ML10/ML45 held-out suites and, separately, on a physical SO-101 robot. Meta-World contributes no source learning or tuning. Appendices D, J, and K give the physical, simulation, and adapter protocols.

Evaluation and comparisons. The primary metric is the budget-weighted anytime success score C_{anytime} (Equation 2). It averages held-out success over checkpoints selected without access to held-out outcomes. We report two evaluation protocols. In common-evidence evaluation, methods receive the same source comparisons. In independent acquisition, each method collects its own source evidence under the same caps. The main predictor baselines are CE-POINT, CE-LOCAL, and CE-U. We also include generic verification, hand-written gates, reward-design methods, and memory-reuse controls. Appendix F provides implementation details for the gates, tuning grids, and adaptations.

Replication and resources. We use 12 independently constructed source states. Each source state/task pair has four target decision trajectories. Each trajectory uses three fresh policy seeds. The source states are the independent replication units. All methods have matched interaction, feedback, inference, and learned-state budgets. Appendix A provides the detailed accounting, sampling, and failure rules.

We organize the results around three questions: whether the method improves transfer, what explains the improvement, and where it remains limited.

5.2 DOES LEARNING VALIDITY BOUNDARIES IMPROVE TRANSFER?

Table 1: Common-evidence performance across six target families (two configurations each). Entries report C_{anytime} for each family. Avg is the equal-weight average across the six families. All methods use matched source evidence and resource caps.

Method	Stack	Contact	Insertion	Articulation	Placement	Sequential	**Avg**
Stateless multi-round trainer	0.24	0.64	0.34	0.42	0.48	0.34	0.41
Adapted agentic memory	0.34	0.73	0.46	0.54	0.53	0.40	0.50
Recipes + generic verifier	0.39	0.74	0.47	0.56	0.53	0.43	0.52
Source-frozen hand-written gate	0.44	0.76	0.52	0.61	0.58	**0.47**	0.56
CE-POINT	0.48	0.79	0.54	0.64	0.59	0.44	0.58
CE-LOCAL	0.48	0.79	0.55	0.64	0.60	0.45	0.59
CE-U	0.49	0.79	0.55	0.65	**0.61**	0.45	0.59
Eureka (Ma et al., 2024a)	0.56	0.71	0.51	0.50	0.48	0.39	0.53
Text2Reward (Xie et al., 2024)	0.60	0.84	0.48	0.52	0.50	0.43	0.56
A-MEM (Xu et al., 2025)	0.36	0.75	0.49	0.57	0.55	0.42	0.52
SELFMEM (Yang et al., 2026)	0.41	0.77	0.51	0.60	0.57	0.44	0.55
MemRL (Zhang et al., 2026b)	0.45	0.76	0.57	0.59	0.44	0.42	0.54
MemSkill (Zhang et al., 2026a)	0.59	0.81	0.52	0.62	0.41	0.43	0.56
Training Contracts (Ours)	**0.63**	**0.85**	**0.62**	**0.69**	0.60	0.45	**0.64**

Table 1 also reports Eureka and Text2Reward as reward-design baselines, together with A-MEM, SelfMem, MemRL, and MemSkill as protocol-matched memory adaptations. These adaptations use the same frozen proposal model, legal intervention interface, and target-time resource constraints rather than reproducing the methods' original task environments (Appendix F).

The Avg column is 0.640 for Contracts, compared with 0.590 for CE-U, 0.585 for CE-LOCAL, and 0.580 for CE-POINT in Table 1. Recipes + generic verifier reaches 0.520. Thus the gain remains against a predictor that receives the same evidence and features and can use uncertainty, abstention, and paired probes. The contract–CE-U difference is 0.050, with a simultaneous interval of $[0.026, 0.074]$ across the 12 source-state constructions.

The advantage also remains when methods acquire their own source evidence. Contracts score 0.615 versus 0.563 for CE-U, a difference of 0.052 with an interval of $[0.026, 0.078]$. The two protocols differ in how source evidence is obtained. Under shared evidence, all methods receive the same source comparisons. Under independent acquisition, each method also bears the consequences and costs of its own evidence-collection decisions. The contract–CE-U contrast remains similar across the two settings. In each protocol, Contracts outperform CE-U in 11 of 12 independent source-state constructions (Appendix G). Uncertainty is computed across these constructions, conditional on the fixed target suite.

5.3 Understanding the Source of Improvement

Mechanism evidence. Table 2 reports both runtime interventions on the learned state and source-state reconstructions. By passing validity-region decisions, hiding confirmation metadata, or hiding counter evidence each degrades performance. Giving CE-U the learned contract regions nearly closes the remaining gap.

Table 2: Ablations and diagnostic interventions on the learned trainer state, including runtime modifications and source-state reconstructions.

Additional diagnostic	C_{anytime}	Difference from 0.640
Training Contracts (Ours), adaptive 20% cap	**0.640**	Reference
Runtime changes to the frozen state		
Bypass learned validity-region decisions	0.520	−0.120
Hide confirmation-effect metadata	0.574	−0.066
Hide counterevidence at retrieval	0.598	−0.042
Disable target probes; reallocate unused budget	0.575	−0.065
Matched predicate permutation, adaptive cap	0.525	−0.115
CE-U with imported learned contract validity regions	0.637	−0.003
Rebuild the source state without a component		
Reconstruct source state without confirmation	0.572	−0.068
Reconstruct with positive-only consolidation	0.561	−0.079

The reconstruction ablations show the same pattern during source-state construction: independent confirmation and retained negative evidence both matter.

Giving CE-U the regions learned by the contract procedure raises its score to 0.637, close to the full method, while leaving the downstream predictor unchanged.

Decision quality at matched coverage. The decision audit evaluates the same 960 intervention contexts. Appendix H reports the fixed-allocation retention and blocking counts.

At exactly 45% coverage, each method accepts 432 cases. Contracts accept 25 known harmful interventions and retain 369 beneficial ones, compared with 49 and 348 for CE-U. The known-harm fractions are 5.8% and 11.3%. Even counting every accepted unresolved case as harmful gives 9.0% for Contracts versus 14.1% for CE-U. Acceptance ranks use predecision scores, and audit labels come from independent continuations. The equal-coverage comparison does not tune thresholds on target outcomes. Appendix H reports the full coverage sweep (Figure 5).

Predicate permutation preserves intervention types, evidence volume, rule complexity, and resource limits while breaking condition–evidence alignment. It accepts 80 harmful cases at 45% coverage and reduces fixed-allocation C_{anytime} from 0.632 to 0.514.

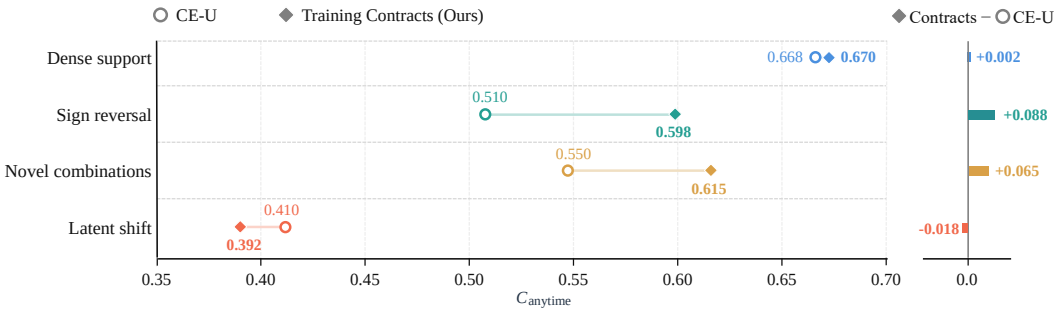


Figure 2: Performance across four reuse regimes: dense support without observed intervention sign reversal; observed intervention sign reversal; novel context combinations composed of supported features; and latent shift not represented in the feature registry.

Probe allocation does not account for the advantage. With probes disabled, contracts still reach 0.575 versus 0.535 for CE-U. At the primary 20% cap, they reach 0.640 versus 0.590. With the same fixed 12% probe allocation for every method, Contracts score 0.632 versus 0.580 for CE-U (Appendix H, Table 12). In the fixed-allocation diagnostic, every method spends the same probe budget even when it would otherwise skip a probe, so the comparison is not driven by different probe expenditure.

5.4 Robustness and Transfer Beyond the Primary Setting

Figure 2 shows that Contracts and CE-U are nearly tied with dense source support: 0.670 versus 0.668. The contract advantage is larger when effects reverse across observed contexts ($+0.088$) or when targets combine supported features in sparsely observed ways ($+0.065$). These are the settings where retaining both positive and negative intervention evidence is most useful in the reported comparisons.

The hidden-shift condition reverses the result: contracts score 0.392 versus 0.410 for CE-U. In the approach-reward example, an unobserved actuator delay makes reuse harmful even though the visible features satisfy the learned rule. The worked example isolates this failure: at the same visible reach frequency $r = 0.89$ and conditional stable completion $h = 0.22$, the independent paired effect changes from $+0.180$ to -0.080 after an unobserved two-control-step actuator delay (Appendix I, Table 17). Thus a matched predicate is only as informative as the effect modifiers represented in its context. Monitoring can stop further loss, but cannot recover interactions already spent.

Transfer across proposal models, learners, and dynamics. Table 3 evaluates the same reuse mechanism under a second proposal backbone, a SAC learner adapter, and shifted mass and friction.

Table 3: Transfer across proposal backbones, learners, and dynamics.

Setting	CE-U	**Training Contracts** (Ours)	Difference
Primary backbone / PPO	0.590	**0.640**	+0.050
Second backbone / PPO	0.550	**0.589**	+0.039
Primary backbone / SAC adapter	0.575	**0.592**	+0.017
Mass/friction shift	0.500	**0.545**	+0.045

The gain is smaller with the SAC adapter but remains positive in all three transfer panels. The positive margins across these panels suggest that the learned validity boundary transfers beyond the primary PPO/backbone setting, while the smaller SAC gain indicates sensitivity to learner-specific adaptation. Together with the latent-shift result above, these results separate changes that preserve the observed intervention context from changes that introduce an unobserved effect modifier. Appendix J gives the corresponding adapter details.

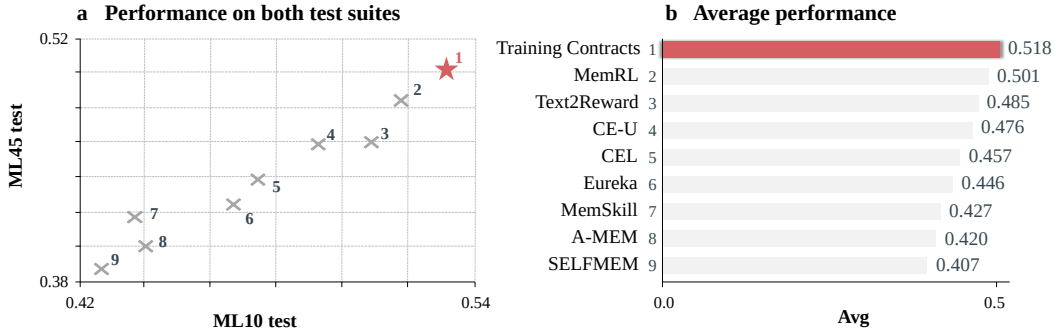


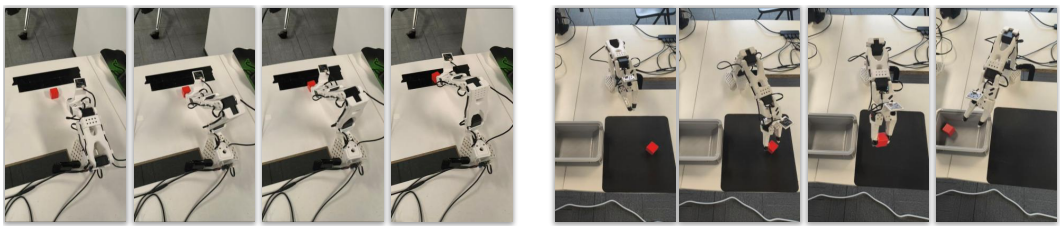
Figure 3: Cross-benchmark transfer to Meta-World. Entries report C_{anytime} on the official ML10 and ML45 held-out test suites. Avg is their arithmetic mean.

Public cross-benchmark transfer on Meta-World. We also evaluate the source-trained trainer with fresh policies on the official ML10/ML45 held-out suites (Yu et al., 2020). No Meta-World task

is used for source learning or tuning, and incompatible edits are rejected. Figure 3 reports the main comparison. Appendix J reports the corresponding mean $\pm$ standard deviation.

5.5 Physical cross-embodiment evaluation

We evaluate transfer from the ManiSkill/Panda source trainer to Cube Pushing and Cube Pick-and-Place on a physical SO-101 manipulator (Figure 4). The source trainer is fixed before this evaluation, and only trainer memory transfers. Each method uses 10 independent training runs per task, each evaluated on six held-out trials. The adapter exposes only semantically compatible diagnostics and interventions. Unsupported edits are rejected (Appendix K).



(a) Cube Pushing **(b)** Cube Pick-and-Place

Figure 4: Physical SO-101 evaluation on Cube Pushing and Cube Pick-and-Place. Each sequence is ordered from left to right.

Table 4: Physical SO-101 results under matched training budgets. For each method and task, success aggregates 60 held-out trials (six per independently trained policy). The last column reports mean C_{anytime} across 10 training runs.

Method	Push Success (%)	Pick-place Success (%)	Avg Success (%)	C_{anytime} (%)
Stateless multi-round trainer	68.3	61.7	65.0	59.8
Recipes + generic verifier	71.7	65.0	68.3	62.5
CE-U	75.0	68.3	71.7	69.1
SELFMEM (Yang et al., 2026)	76.7	68.3	72.5	67.2
MemSkill (Zhang et al., 2026a)	68.3	75.0	71.7	69.2
MemRL (Zhang et al., 2026b)	81.7	71.7	76.7	72.3
Training Contracts (Ours)	**86.7**	**76.7**	**81.7**	**78.8**

Training Contracts obtain 81.7% average final success and a 78.8% anytime score, compared with 71.7% and 69.1% for CE-U and 76.7% and 72.3% for MemRL (Table 4). The six trials per policy assess the same trained learner. The 10 training runs are the independent units. No SO-101 observations enter source discovery or confirmation. The gains in both final success and anytime performance suggest that the transferred trainer memory benefits learning under the matched physical interaction budget. Since target policies start from scratch, this comparison supports reuse of intervention knowledge across embodiments. Its scope remains limited to the compatible diagnostics and interventions exposed by the adapter; it does not establish validity for arbitrary hardware changes or unobserved effect modifiers.

6 Conclusion

Training Contracts use paired intervention evidence to learn where an intervention should be reused, rejected, or tested further. On the primary simulation suite, learning these validity regions improves transfer over matched conditional-effect and memory controls. The source-trained trainer also transfers to the official Meta-World benchmarks and to a physical SO-101 evaluation. These results suggest that reusable training experience benefits from explicit validity boundaries rather than context-free reuse of past interventions. When an important shift is missing from the observed context, however, the learned boundary can fail, motivating richer context representations and broader real-world evaluation settings under diverse deployment conditions.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, clarity and organization, literature search, figure preparation, and limited code and LaTeX editing. All research decisions, experimental protocols, implementations, reported numerical results, and scientific conclusions were determined and verified by the authors.

ETHICS STATEMENT

This work studies the reuse of training interventions in simulation and on a physical SO-101 manipulator. It does not involve human subjects, personally identifiable information, or private datasets.

REPRODUCIBILITY STATEMENT

Section 4 defines the method and target-time decision rule. The appendices provide implementation pseudocode, lifecycle details, environments and target configurations, intervention and feature definitions, controller and learner settings, baseline adaptations, sampling procedures, probe and resource budgets, randomization and seed structure, statistical analysis, and the decision-audit protocol.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016.

Anastasios N. Angelopoulos and Stephen Bates. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *arXiv preprint arXiv:2107.07511*, 2021. URL https://arxiv.org/abs/2107.07511.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal Risk Control. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016. doi: 10.1073/pnas.1510489113.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized Random Forests. *The Annals of Statistics*, 47(2):1148–1178, 2019.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying Generalization in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289, 2019.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056, 2020.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779*, 2016. URL https://arxiv.org/abs/1611.02779.

Ran El-Yaniv and Yair Wiener. On the Foundations of Noise-free Selective Classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1515–1528, 2018.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, 2016.

Yonatan Geifman and Ran El-Yaniv. Selective Classification for Deep Neural Networks. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated Curriculum Learning for Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1311–1320, 2017.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In *International Conference on Learning Representations (ICLR)*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population Based Training of Neural Networks. *arXiv preprint arXiv:1711.09846*, 2017. URL `https://arxiv.org/abs/1711.09846`.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized Level Replay. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950, 2021.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning Representations for Counterfactual Inference. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pp. 3020–3029, 2016.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023. doi: 10.1613/jair.1.14174.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019. doi: 10.1073/pnas.1804597116.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024a.

Yecheng Jason Ma, William Liang, Hung-Ju Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. DrEureka: Language Model Guided Sim-To-Real Transfer. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024b. doi: 10.15607/RSS.2024.XX.094.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-Student Curriculum Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, 2020. doi: 10.1109/TNNLS.2019.2934906.

Meta. Llama-3.1-8B-Instruct: official model card and checkpoint repository. Hugging Face repository `meta-llama/Llama-3.1-8B-Instruct`, 2024. URL `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`. Accessed 7 September 2026.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv preprint arXiv:1803.02999*, 2018. URL `https://arxiv.org/abs/1803.02999`.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing Generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:1810.12282*, 2018. URL `https://arxiv.org/abs/1810.12282`.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pp. 737–746, 2016.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic Curriculum Learning for Deep RL: A Short Survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4819–4825, 2020. doi: 10.24963/ijcai.2020/671.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340, 2019.

Yaniv Romano, Matteo Sesia, and Emmanuel Candès. Classification with Valid and Adaptive Coverage. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL `https://arxiv.org/abs/1707.06347`.

Uri Shalit, Fredrik D. Johansson, and David Sontag. Estimating Individual Treatment Effect: Generalization Bounds and Algorithms. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3076–3085, 2017.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe Exploration for Optimization with Gaussian Processes. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 997–1005, 2015.

Stefan Wager and Susan Athey. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018. doi: 10.1080/01621459.2017.1319839.

Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016. URL https://arxiv.org/abs/1611.05763.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward Shaping with Language Models for Reinforcement Learning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic Memory for LLM Agents. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-Gradient Reinforcement Learning. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 2396–2407, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2715518c875999308842e3455eda2fe3-Abstract.html.

Zhongwen Xu, Hado van Hasselt, Matteo Hessel, Junhyuk Oh, Satinder Singh, and David Silver. Meta-Gradient Reinforcement Learning with an Objective Discovered Online. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Shu Yang, Junchao Wu, Derek F. Wong, and Di Wang. SelfMem: Self-Optimizing Memory for AI Agents. *arXiv preprint arXiv:2607.03726*, 2026. URL https://arxiv.org/abs/2607.03726.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and Efficient Large-scale Reinforcement Learning via Macro-to-Micro Flow Transformation. *arXiv preprint arXiv:2509.15965*, 2025. URL https://arxiv.org/abs/2509.15965.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100, 2020.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. MemSkill: Learning and Evolving Memory Skills for Self-Evolving Agents. *arXiv preprint arXiv:2602.02474*, 2026a. URL https://arxiv.org/abs/2602.02474.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954*, 2025. URL https://arxiv.org/abs/2505.22954.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-Evolving Agents via Runtime Reinforcement Learning on Episodic Memory. *arXiv preprint arXiv:2601.03192*, 2026b. URL https://arxiv.org/abs/2601.03192.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

A Detailed Experimental Protocol

A.1 Two protocols with different attribution

Common evidence gives every method the same source comparisons within each lineage, including failures and confirmation forks. **Independent acquisition** lets each principal comparator collect its own source trajectories under the same caps, including the costs of tuning, rejected candidates, discovery, confirmation, compilation, and sentinel checks. Source initialization blocks and target panels are coupled across methods, while scores and costs are recorded separately for each protocol.

A.2 Task roles and independent replication

Source anchors are PickCube-v1, PokeCube-v1, LiftPegUpright-v1, and PlaceSphere-v1. StackCube-v1, PushCube-v1, and PullCube-v1 are development tasks. The primary suite contains two custom configurations in each of six families: Stack, Contact, Insertion, Articulation, Placement, and Sequential. Appendix D gives all geometry, tolerances, dynamics, reset rules, and success predicates.

The cross-benchmark panel uses the official Meta-World ML10 and ML45 held-out test suites (Yu et al., 2020) with their Sawyer manipulator and MuJoCo dynamics. Meta-World is used only for evaluation. Target policies start from scratch, and the source trainer exposes only diagnostics and interventions whose semantics carry across benchmarks. Unsupported edits are rejected.

We use 12 independent source-state constructions. Each lineage/task pair has four target decision trajectories, and each trajectory has three fresh policy seeds. This yields $12 \times 12 \times 4 = 576$ trainer trajectories and 1,728 policy paths per method at the final source state, before source runs or probes. The 12 source constructions are the replication units for uncertainty.

A.3 Budgets, evaluator, and inference limits

For simulation experiments, the per-policy-path cap is B = 20.48 million environment interactions. Each path has four decision windows, at most eight proposal calls including repairs, and a 20% maximum probe share. Decisions occur at $0, 0.25B, 0.50B, 0.75B$. Held-out scores are committed at $b_j = jB/8$, $j = 1, \ldots, 8$. Each decision uses 64 development episodes per seed. Held-out scoring uses 128 episodes per seed at each boundary. Episodes have at most 200 control steps. Development interactions count toward B, whereas held-out evaluation is separately metered on the same schedule for every method. A paired probe reserves at most $0.05B$ per window. The fixed-12% diagnostic uses $0.03B$ per window in a separate ledger.

Per-trajectory limits are eight calls, 131,072 input tokens, 8,192 generated tokens, and 64 MiB of target-accessible learned state. Each call is capped at 16,384 input and 1,024 generated tokens. Failed proposals and repairs remain charged.

A.4 Strong baselines and tuning

The conditional-effect controls receive the same compiler features, intervention identities, positive/negative/null comparisons, root-level split assignments, and source tuning budget as contracts. CE-Point predicts a conditional mean. CE-U adds nonlinear ensembles, calibrated residual uncertainty, out-of-support abstention, and the same paired probes. CE-Local uses local evidence neighborhoods and uncertainty. Model and gate selection use source-only data.

Memory controls include A-MEM-style and SelfMem-style trainer-memory adaptations, plus MemRL and MemSkill variants specified in Appendix F.4. These methods share the source records, predecision feature registry, frozen proposal model, learned-state cap, retrieval/token/call limits, and target-time probe opportunities, but do not receive contract predicates or imported contract validity regions. Additional controls are the stateless trainer, generic verifier, source-frozen hand-written gate, offline compilation, predicate permutation, and the bridge diagnostic. Appendix F provides implementation details and tuning grids.

A.5 FAILURE-INCLUSIVE STATISTICS

All assigned trajectories remain in the denominator. Invalid proposals consume their allowance, fallback checkpoints are scored, and terminal failures without a usable checkpoint receive zero. Hardware or infrastructure outages are handled with the paired-block rerun rule.

The two primary contrasts are contracts minus CE-U under common evidence and independent acquisition. Task-balanced contrasts are formed within source lineages. Simultaneous Student-t intervals use the original three-contrast Bonferroni allocation. Because only two contrasts are reported here, these intervals are conservative. The estimand is conditional on the twelve-task suite.

B DECISION-LEVEL ANALYSIS

B.1 INTERVENTION-LEVEL REGRET

Let T denote the current learner checkpoint and $T \oplus u$ its continuation under intervention u. For an audit context x, legal intervention u, comparator, and horizon, let

$$\Delta(x, u) = \mathbb{E}[U(T \oplus u) - U(T) \mid x]. \tag{9}$$

Let $g(x, u) \in \{0, 1\}$ be a promotion decision fixed before independent audit execution, including any earlier charged development probe. Relative to an oracle that promotes when $\Delta > 0$,

$$\mathcal{R}(g) = \mathbb{E}[\Delta_+ - g\Delta] = \mathbb{E}[|\Delta| \, \mathbf{1}\{g \neq \mathbf{1}[\Delta > 0]\}] \, . \tag{10}$$

The identity separates missed beneficial transfers from harmful promotions. It concerns a fixed local comparison: interventions change later contexts, and development utility can differ from final utility.

Proposition 1 (Decision-level regret identity)**.** *For an audit context with expected paired effect Δ and a binary promotion decision g fixed before independent audit execution,*

$$\Delta_+ - g\Delta = |\Delta| \mathbf{1}\{g \neq \mathbf{1}[\Delta > 0]\}.$$

The same identity holds in expectation for randomized decisions, including their randomness in the expectation.

Proof. When $\Delta > 0$, the left side equals $(1 - g)\Delta$, which is nonzero exactly when a useful intervention is rejected. When $\Delta < 0$, it equals $g|\Delta|$, which is nonzero exactly when a harmful intervention is promoted. Both sides vanish for $\Delta = 0$. □

The comparator, continuation horizon, and context distribution are set before effect evaluation. Development probes and audit continuations use independent randomness and separately recorded outcomes, so the audit label is not reused in the intervention decision.

B.2 SCOPE OF SOURCE CONFIRMATION

Source confirmation characterizes the finite validity region represented by the observed source contexts. Extending that evidence to a shifted target distribution requires stability of $\Delta(x, u)$ and a context representation that captures the relevant effect modifiers. The latent-delay case in Table 17 illustrates what happens when an important modifier is absent from the observed context.

C RUNTIME AND LEARNING CONFIGURATION

C.1 TRAINER STATE AND TRANSFER RESTRICTIONS

A training problem is $P = (E, \mathcal{D}^{\text{train}}, \mathcal{D}^{\text{dev}}, \mathcal{D}^{\text{test}}, J_P, \mathbf{B}_P)$ and a trainer version is

$$\mathcal{A}_k = (G, \rho, A_k), \qquad A_k = (M_k, R_k, W_k, \mathcal{I}_k, S_k). \tag{11}$$

The proposal model G, orchestration ρ, legal action schema, and held-out evaluation harness $\mathcal{H}$ are shared across the experiment. M stores trial evidence, R diagnostic knowledge, W workflows, $\mathcal{I}$

intervention templates, and S screening and allocation knowledge. Contracts connect these collections within the same bounded trainer-state interface.

Each target task starts with fresh policy parameters, optimizer state, replay buffers, and checkpoints. The source state A_k remains fixed throughout a target panel. A transient history $h_{P,t}$ supports diagnosis within one target problem and is discarded afterward. Checkpoint continuation and rollback are confined to the current target lineage.

C.2 Models, inference, and hardware

The primary proposal model is `Qwen/Qwen2.5-32B-Instruct` (Yang et al., 2024). The second backbone is `meta-llama/Llama-3.1-8B-Instruct` (Meta, 2024). Inference uses the same model weights and chat templates throughout each backbone panel, with BF16 without quantization, temperature 0.2, top-$p = 0.95$, at most 1,024 generated tokens, and at most 16,384 input tokens per prompt. Sampling and repair settings are shared across methods within a panel.

The hardware profile allocates one A100-80GB worker per active policy path and a two-A100-80GB proposal server. Up to three policy paths from one trainer trajectory may run concurrently. GPU accounting uses allocated rather than utilization-scaled time.

Table 5: Resource caps used in the primary evaluation.

Resource	Experimental cap / rule
Simulation target interactions	20,480,000 per policy path, including development and probe branches
Target policy paths	Three per trainer trajectory. One shared proposal sequence
Target calls / generated tokens	Eight calls including repairs / 8,192 total generated tokens
Target input tokens	131,072 total. 16,384 per call
Retrieved memory slice	At most 8,192 tokens per call, included in the input cap
Target-accessible learned state	64 MiB total, including prediction weights, evidence summaries, indices, and contracts
Active contracts / retrieved candidates	At most 32 active or restricted contracts / at most eight candidates per window
Target GPU ceiling	60 allocated GPU-hours per three-seed trainer trajectory
Source interactions	16.384 million per independently constructed source lineage
Source calls / input / output tokens	1,024 calls / 16,777,216 input / 1,048,576 output, including tuning and consolidation
Source GPU ceiling	64 allocated GPU-hours per constructed lineage
Source allocation	30% discovery, 40% confirmation, 10% calibration, 10% tuning, 10% sentinel

The source allocation in Table 5 includes checkpoint-root generation, failed branches, and evaluations in the corresponding split. Candidates below the support requirement remain unconfirmed.

C.3 Primary PPO learner

We use PPO (Schulman et al., 2017) with the shared configuration in Table 6.

Budget-ending partial rollouts are masked, and no additional complete rollout is collected after the interaction cap.

Legal optimization edits include learning-rate multipliers in $\{0.25, 0.5, 1, 2\}$. Entropy coefficients are selected from $\{0, 0.005, 0.01, 0.02\}$ and clipping ratios from $\{0.1, 0.2, 0.3\}$. Legal reward edits modify allowlisted auxiliary coefficients in $\{0, 0.25, 0.5, 1, 2\}$ relative to the base coefficient, without changing task-success code, action semantics, observation permissions, or termination. A curriculum edit selects one of three source-defined reset-difficulty levels. Invalid or unsupported changes fail common validation. A compound edit is an explicit bundle with evidence for that bundle.

The fallback is the unchanged current learner with the default exploration setting. Rollback restores the complete target-local checkpoint state from a checkpoint created before the current decision.

Table 6: PPO configuration used in the primary evaluation.

PPO setting	Experimental value
Observation / control	State observations. Bounded end-effector delta pose plus gripper command
Actor / critic	Separate MLPs, two hidden layers of 256 units, tanh activations
Action distribution	Diagonal Gaussian with tanh transform. Initial log standard deviation -0.5. Correct transform Jacobian in log probability
Parallel environments / rollout	1,024 / 32 control steps, or 32,768 transitions per complete rollout
Optimizer / learning rate	Adam / 3×10^{-4} before a legal trainer edit
Discount / GAE	0.99 / 0.95
Clip ratio / value coefficient	0.2 / 0.5
Epochs / minibatch	Four / 4,096 transitions
Entropy coefficient / gradient clip	0.005 / global norm 0.5
Advantage normalization	Per rollout. Statistics do not mix independent policy seeds
Observation normalization	Training-only running mean/variance. Freeze for development and held-out evaluation. Clip normalized observations to $[-5, 5]$
Time limits	Bootstrap timeout truncations. Do not bootstrap true terminal failures
Reward / checkpoints	Keep task reward scale fixed. Save actor, critic, optimizer, normalization, and random-generator state together

C.4 Second learner and semantic adapters

The second learner is SAC (Haarnoja et al., 2018). It uses a separate two-layer 256-unit ReLU actor and twin critics. The discount is 0.99, and the target-update coefficient is 0.005. Adam uses learning rate 3×10^{-4} with batch size 256. Replay capacity is 10^6 transitions, with 10,240 initial random interactions. The actor uses a tanh-Gaussian distribution with automatic entropy adjustment and target entropy equal to minus the action dimension. The learner performs 32 gradient minibatches per 1,024 newly collected transitions, with compute metered separately. Replay and optimizer state remain target-local.

Only semantically matched intervention types transfer to SAC. PPO-specific clipping edits remain unsupported. Learner-specific diagnostics use a versioned adapter. Missing features follow the common uncertain-support path. The portability panel uses the same interaction-normalized metric as the primary study.

C.5 Checkpoint windows, probes, and promotion

At each of the four main windows, the trainer may issue one candidate and at most one repair. Together, these calls must respect the eight-call ceiling. Source-guided retrieval and gate evaluation are deterministic given state, observation, and the recorded random stream. The retry policy is shared across methods.

An adaptive probe reserves at most $1{,}024{,}000 = 0.05B$ interactions per path per window, split equally between candidate and comparator. Each branch's reservation includes up to 64 development episodes of at most 200 steps. The branch trains with the remaining allocation and then evaluates without updating policy or normalization statistics. The two branches share initialization and exogenous reset streams, not future outcomes. The fixed-12% diagnostic replaces the per-window reservation with $614{,}400 = 0.03B$. It executes the same scheduled allocation for every method, including controls that would otherwise avoid a probe. This fixed allocation is used only for the diagnostic comparison, not for the primary deployed policy.

Promotion requires a candidate-minus-comparator development utility of at least 0.02 after the shared validity checks. The utility is averaged over the three policy seeds, and no individual seed may fall below -0.05. Otherwise, the comparator remains the incumbent. The same selection rule applies to every method. Confirmatory audit runs use new randomness. During a reused window, a later scheduled development comparison may trigger rollback if development success drops by more than

0.05 relative to the stored incumbent reference. Every method uses the same monitor and schedule, and no extra evaluation is added outside the budget.

The public development utility is success frequency under the public goal predicate at the fixed continuation horizon. The held-out evaluator uses the same goal semantics with independent held-out resets. Learning-curve summaries and diagnostics use only information available before the current decision. At the first window, unavailable statistics are explicitly missing and the fallback is used when required by the gate.

C.6 Source roots, randomization, and split leakage

Before training, each root is assigned immutable metadata specifying its lineage, source stage, task, root index, and split role. All descendants remain in that split. Rollout episodes, policy seeds, and late checkpoints from one root cannot be partitioned across discovery and confirmation. A predicate discovered after confirmation feedback requires a fresh confirmation allocation, while the previous failed candidate remains in the archive.

Source orders are generated independently from the 24 permutations of the four anchors using recorded random streams. Duplicate orders are allowed. Independent lineages use independent source roots and proposal randomness. Method pairing couples exogenous environment initialization and policy-seed streams at the block level. The coupling is defined by these streams rather than by matching integer seed values alone. Counter-based environment/reset streams preserve the intended pairing when branches consume different numbers of random draws.

D Target Specifications and Feature Registry

D.1 Primary task suite and configuration identities

Table 7 specifies the twelve-configuration primary suite. Names beginning `ref/` are local configuration identifiers, not public ManiSkill task registrations. The local prefixes F1–F6 correspond to the family labels used in Table 1: Stack, Contact, Insertion, Articulation, Placement, and Sequential. The task wrappers and assets were frozen before target execution.

The primary ManiSkill-based experimental scene uses a Franka Emika Panda seven-degree-of-freedom manipulator with a parallel-jaw gripper. The scene has a horizontal tabletop, state observations, and a 200-step horizon. Control runs at 20 Hz with 200 Hz simulation. A delta-pose action changes translation by at most 0.02 m per axis and rotation by at most 0.10 rad per axis per control step. A normalized gripper command is appended to the action. All methods use identical action semantics. Default dynamic randomization is mass multiplier $[0.8, 1.2]$ and friction coefficient $[0.6, 1.0]$, sampled uniformly at reset. Cubes have nominal edge 0.04 m and mass 0.06 kg unless overridden. Spheres have radius 0.02 m and mass 0.06 kg. Coordinates are defined in a published tabletop frame.

Training resets sample object positions from a 0.20 m square and initial yaw from $[-\pi, \pi]$. Feasibility rejection sampling allows at most 100 attempts. Development and held-out tests use disjoint random streams from the same distribution. Shift panels use explicitly changed distributions. The task-configuration specification fixes contact topology, nominal inertias, collision meshes, joint limits, and controller gains independently of target outcomes, complementing the higher-level geometry and goal descriptions.

Table 7: Primary target-task definitions.

Local ID / family	Geometry, goal, and distinguishing requirement
`ref/F1-A` Stack-and-release	Stack a 0.04 m cube on a fixed 0.05 m support cube. Horizontal center error ≤ 0.02 m. Height error from supported contact ≤ 0.01 m. Released. Linear speed ≤ 0.03 m/s and angular speed ≤ 0.3 rad/s.

Local ID / family	Geometry, goal, and distinguishing requirement
`ref/F1-B` Stack-and-release	Same goal type with a 0.03 m movable cube and 0.04 m support. Horizontal tolerance 0.01 m and height tolerance 0.0075 m. Initial support location and object yaw vary independently. This is a within-family precision variation, not a new task family.
`ref/F2-A` Directional contact	Push a 0.04 m block 0.15 m along the public positive tabletop x direction. Final position error ≤ 0.02 m, height within 0.01 m of the tabletop, speed ≤ 0.03 m/s. No lifted or attached-object solution.
`ref/F2-B` Directional contact	Pull a ridged block 0.15 m along the public negative tabletop x direction using finger contact without an attached tool. Same terminal tolerances. The ridged collision asset and permitted contact geometry are part of the task-wrapper specification.
`ref/F3-A` Insertion/alignment	Insert a cylindrical peg of radius 0.010 m at least 0.03 m into a socket with 0.003 m radial clearance. Axis error $\leq 5^\circ$, lateral error ≤ 0.003 m. All conditions hold at the terminal dwell.
`ref/F3-B` Insertion/alignment	Insert a square peg into a square slot with 0.0015 m per-side clearance, depth ≥ 0.03 m, lateral error ≤ 0.0015 m, and yaw error $\leq 3^\circ$. The slot remains fixed during an episode.
`ref/F4-A` Articulated object	Open a one-degree-of-freedom drawer by at least 0.12 m from its closed position. Final joint speed ≤ 0.02 m/s. Handle geometry, damping, and joint friction are fixed by the wrapper plus declared reset randomization.
`ref/F4-B` Articulated object	Open a hinged panel to at least 60° with final angular speed ≤ 0.10 rad/s. Joint limits prevent wrapping the angle. Only handle/gripper contacts count as legal manipulation.
`ref/F5-A` Precision placement	Place a 0.02 m-radius sphere into a shallow cradle with center error ≤ 0.015 m, released, speed ≤ 0.02 m/s, and ten consecutive stable steps. Cradle collision geometry is part of the declared asset.
`ref/F5-B` Precision placement	Place a 0.03 m cube in a planar target with center error ≤ 0.01 m, yaw error $\leq 5^\circ$, released, and ten stable steps at speed ≤ 0.02 m/s.
`ref/F6-A` Sequential relocation	Place object A then object B into two public target regions 0.15 m apart. Each has 0.02 m position tolerance and a released five-step dwell. A must satisfy its dwell before B's completion is credited. Both remain placed at termination.
`ref/F6-B` Sequential relocation	Relocate one cube through two public waypoints 0.12 m apart in order, with released five-step dwell at the first and ten-step dwell at the second. The evaluation terminal condition requires the complete ordered event history, not only final proximity.

Success must hold for the last five control steps unless a longer dwell is specified. "Released" means no gripper–object contact and gripper opening at least 80% of its maximum.

The transfer setting includes held-out configurations, changed goal conditions, and differences in family structure where applicable. The stack-and-release and directional-contact families share conceptual structure with the stack/push/pull development tasks. The panel therefore mixes within-family transfer with more substantial family shifts rather than using a uniformly family-held-out design.

D.2 Versioned predecision feature registry

The registry contains public categorical metadata, normalized scalar diagnostics, and missing-value indicators. Categorical fields include action-controller family, learner adapter, intervention type, and public goal form. Numeric fields are computed from the most recent declared development panel and the preceding two available windows. Protected outcomes are not used.

Development success semantics are public. Held-out reset seeds, held-out outcomes, and hidden shift labels are excluded from the feature registry. Constant-feature and missing-value cases use the same definitions for every method.

Table 8: Predecision feature definitions and missing-value semantics.

Feature	Computation / unit	Missing rule
Reach frequency r	Fraction of development episodes with object–end-effector distance ≤ 0.04 m for two consecutive steps. Unitless	Unknown if no panel
Conditional stable completion h	Fraction of reaching episodes that also satisfy the public stable-completion condition. Unitless	Unknown when no episode reaches
Development success	Fraction satisfying the public task goal under frozen evaluation normalization	Unknown before first panel
Progress slope	Change in development success divided by million charged interactions between the two latest panels	Unknown with fewer than two panels
Action saturation	Fraction of scalar action components with absolute normalized magnitude ≥ 0.95	Unknown without action trace
Action variability	Mean per-dimension standard deviation over the latest training rollout. Normalized-action units	Unknown without rollout
Critic residual	Mean absolute TD residual divided by source-frozen scale for the declared learner adapter	Unknown for incompatible adapter
Budget fraction	Charged interactions divided by B, in $[0, 1]$	Always present
Source-support distance	Distance to declared source support in normalized observed features, with missingness handled explicitly	Unknown with no compatible support

Continuous thresholds are drawn from discovery quantiles and serialized with their units and source ranges. Missing-value bits are available to every predictor and the contract executor. Missing h values use a common representation across methods. Predictor imputations, when used, rely on source-only statistics together with the missingness indicator.

E Contract Compiler, Confirmation, and Executable Semantics

E.1 Contract record and admission rule

Equation 3 is the executable contract record. Evidence is attached to the intervention or intervention bundle that generated it, with supporting and contradicting provenance retained through revision and merging.

For stratum c, $\Delta_e(c) = \widehat{U}_{\mathrm{dev}}(T \oplus u.c) - \widehat{U}_{\mathrm{dev}}(T.c)$. Positive admission follows

$$\begin{aligned}\mathrm{Admit}^{+}(e) = \; & \mathrm{Valid}(e) \wedge \mathrm{Prov}(e) \wedge \mathrm{Disjoint}(e) \wedge \mathrm{Support}(e) \\ & \wedge \left[\min_{c \in \mathcal{C}_e} \mathrm{LCB}_c(\Delta_e) \geq -\epsilon\right] \wedge \left[\max_{c \in \mathcal{C}_e} \mathrm{LCB}_c(\Delta_e) > \eta\right]. \end{aligned} \tag{12}$$

If the validity boundary is revised after discovery, the revised predicate is treated as a new candidate and must be confirmed on freshly allocated evidence.

State updates use Add, Revise, Merge, Reweight, and Deprecate. Merging preserves provenance and contradictory comparisons. A sentinel comparison on source data can admit, restrict, or roll back a candidate state within the confirmed source scope. Active-state, archive, retrieval, and inference limits are shared with memory and prediction baselines. Learned predictor weights count toward the state budget. Raw trace archives used for offline analysis are not accessible during target execution.

Experimental configuration. The compiler uses at most three atoms per clause and two clauses per region. Discovery quantiles are $0.1, 0.2, \ldots, 0.9$. At most eight candidate contracts are considered per source stage. Four stages and at most two tested strata per candidate give at most 64 candidate–stratum tests. Each tested stratum requires at least 24 independent checkpoint-root blocks, averaging three paired policy seeds inside a block. We use $\gamma = 2$, $\tau = 0.002$, $\epsilon = 0.01$, and $\eta = 0.02$ on normalized utility.

E.2 FINITE GRAMMAR AND DISCOVERY SELECTION

An atom is an allowlisted numeric comparison or categorical equality. A clause is a conjunction of at most three atoms. A region is a disjunction of at most two clauses. Numeric threshold candidates are the nine discovery quantiles at $0.1, 0.2, \ldots, 0.9$, with ties collapsed to a single serialized threshold. A supported clause requires at least 24 discovery root blocks. Canonical serialization breaks score ties. Candidate edits outside the legal schema are rejected before any learned gate is evaluated.

At each source stage, at most eight complete contract candidates reach confirmation, with at most two tested strata per candidate. Candidate discovery may use the frozen proposal model, but final predicates must compile into the finite typed grammar. Multiple related clauses remain within the same candidate accounting. Refining a predicate after confirmation creates a new candidate and consumes a remaining slot and fresh root allocation.

The source-side configuration grids are $d \in \{1, 2, 3\}$ atoms, $\gamma \in \{1, 2, 4\}$, $\tau \in \{0, 0.002, 0.005\}$, $\epsilon \in \{0, 0.01, 0.02\}$, and $\eta \in \{0.01, 0.02, 0.04\}$. The source-side search evaluates at most 24 configurations, and the tuning ledger includes both selected and rejected settings. Target outcomes are not used in this search.

One-factor parameter sensitivity. We vary one discovery or confirmation parameter at a time while keeping the remaining settings fixed. Table 9 reports the resulting primary common-evidence score.

Table 9: One-factor sensitivity of Training Contracts to the discovery and confirmation parameters. The default configuration is $\gamma = 2$, $\tau = 0.002$, $\epsilon = 0.01$, and $\eta = 0.02$. Each non-default row changes only the listed parameter.

Setting	$C_{anytime}$	Difference from default
Default	0.64	–
$\gamma = 1$	0.65	+0.01
$\gamma = 4$	0.63	-0.01
$\tau = 0$	0.66	+0.02
$\tau = 0.005$	0.64	0.00
$\epsilon = 0$	0.62	-0.02
$\epsilon = 0.02$	0.66	+0.02
$\eta = 0.01$	0.62	-0.02
$\eta = 0.04$	0.60	-0.04

E.3 CONFIRMATION ESTIMATOR AND MULTIPLE COMPARISONS

For independent checkpoint-root block i, let d_i be the mean of the three seed-paired development-utility differences. With $n \geq 24$ root blocks, the working interval used is

$$L = \bar{d} - t_{n-1,1-\alpha_*} \frac{s_d}{\sqrt{n}}, \qquad U = \bar{d} + t_{n-1,1-\alpha_*} \frac{s_d}{\sqrt{n}}, \tag{13}$$

$$\alpha_* = \frac{0.05}{2 \times 64} = 0.000390625. \tag{14}$$

The allocation covers both directions and all 64 candidate–stratum slots, even when fewer candidates are submitted. Candidates are selected before their confirmation blocks are observed, and the union bound permits dependence across candidates when the individual tests are valid. The Student-t interval is a parametric admission criterion and assumes independent normal root-level effects for exact coverage. Table 16 reports the corresponding arithmetic.

Positive admission uses Equation 12. A contradicted region is supported when its upper bound is below $-\eta$, and unresolved or underpowered strata remain unresolved. Direct reuse requires an active contract, a matched stratum with $L > \eta$, a compatible controller/learner adapter, and observed numeric features within the supported source ranges. Restricted contracts and noninferiority-only strata follow the uncertain path.

E.4 SENTINEL REGRESSION CHECK

After a source update, the candidate and incumbent are compared on a sentinel panel with fresh policy seeds using paired task-balanced differences under the same budget. Reject or restrict an update if any sentinel-family difference is below -0.03 or if the mean is below -0.01. A restricted contract retains its frozen predicates but cannot directly reuse an intervention. It may invoke a complete paired probe when budget permits and otherwise abstains. Sentinel trials, failed candidate states, rollback decisions, and repeated accesses are recorded in the source ledger. Target evaluation uses a separate held-out panel.

E.5 THREE-VALUED PREDICATES AND BUDGET-SAFE EXECUTION

```
# Pseudocode for the implemented decision rule.
def decide(contract, observation, ledger, support):
if not common_guard.valid(contract.intervention):
return REJECT_INVALID
if contract.status not in {ACTIVE, RESTRICTED}:
return common_fallback(observation, ledger)

minus = eval_three_valued(contract.contraindication, observation)
plus = eval_three_valued(contract.applicability, observation)
if minus is TRUE:
return ABSTAIN

# UNKNOWN never silently becomes FALSE.
supported = (
support.compatible_adapter
and support.independent_roots >= 24
and support.current_stratum_lower_bound > 0.02
and support.inside_frozen_feature_ranges
)
if contract.status == ACTIVE and minus is FALSE and plus is TRUE and
    supported:
return REUSE_WITH_COMMON_MONITOR

# Reserve candidate + comparator + their development episodes.
if ledger.can_reserve_complete_pair(contract.probe):
return PAIRED_PROBE
return ABSTAIN
```

Rejected and abstained interventions retain their source evidence, target probes affect only the current task trajectory, and the source state does not change during a target panel. State digests are recorded before and after each panel.

F STRONG CONDITIONAL-EFFECT AND MEMORY CONTROLS

F.1 MATCHED INFORMATION AND CAPACITY

Every comparator sees the same feature registry and intervention identity. Parameter encoding and paired comparison fields are also matched in the common-evidence protocol. A record includes the context before the edit, comparator, continuation horizon, utility definition, independent-root identity, and seed-pair effects. It also records whether the outcome is positive, negative, practically neutral, or unresolved. Root identity is used for splitting and dependence accounting. Hidden target labels are never model inputs.

A predictor may fit on discovery and disclosed confirmation comparisons. It then calibrates on separate calibration roots, giving it more fitting flexibility than a contract's frozen discovery boundary. Target labels are withheld from every method. The methods share counterexample access, prompt limits, target archives, and the 64 MiB learned-state cap. The cap includes predictor weights, calibration buffers, retrieval indices, and text state.

F.2 MEAN AND UNCERTAINTY-AWARE PREDICTION

CE-POINT selects a nonlinear predictor from 24 source-only fitting configurations. Twelve use gradient-boosted regression with depth $\{3, 6\}$, tree count $\{64, 128, 256\}$, and learning rate $\{0.03, 0.1\}$. The other twelve use two-layer MLPs with width $\{64, 128\}$, learning rate $\{3 \times 10^{-4}, 10^{-3}\}$, and weight decay $\{0, 10^{-4}, 10^{-3}\}$. Numerical features use source-only standardization and missingness bits. Intervention parameters are normalized by legal ranges. Categorical fields use fixed one-hot encodings. Five root-grouped folds select fitting settings without mixing descendants of a root.

CE-U uses the same candidate family with five root-bootstrap ensemble members. Fit a mean $\hat{\mu}(x, u)$ and scale $s(x, u)$, then compute held-out source-calibration residuals $|d - \hat{\mu}|/(s + 10^{-3})$. Its lower score is $\hat{\mu} - q(s + 10^{-3})$, where q is a source-calibrated residual quantile. The candidate residual quantiles are $0.80, 0.90, 0.95, 0.975$. All evaluated settings and fitting costs are logged. The point predictor and uncertainty-aware predictor may both probe when their gate is uncertain. The latter also marks an input unsupported when its compatible-neighbor distance exceeds the source-calibration 95th percentile.

Model fitting configurations and gate thresholds are selected on source splits. The reuse gate is selected from the threshold set $\{0.01, 0.02, 0.03, 0.04\}$. With 24 fits and four reuse-threshold levels, up to 96 offline policy evaluations are possible and counted in the tuning budget. The same source tuning ledger caps the contract family's candidate evaluations. At most two finalists per method receive full source-only online validation after the cheap screen. The finalist with the highest source validation utility is selected. Ties are broken using the source-side random stream. The selected threshold is then held fixed for target evaluation.

At target time, let $L(x, u) = \hat{\mu}(x, u) - q(s(x, u) + 10^{-3})$ and let η_{CEU} denote the selected reuse threshold. A compatible in-support candidate is directly reused when $L(x, u) > \eta_{\mathrm{CEU}}$. An in-support candidate that does not pass the direct-reuse threshold may use the common paired probe when a complete probe fits the remaining budget. Otherwise, the method abstains. Inputs marked out of support abstain. For CE-POINT, which has no predictive interval, let η_{CEP} be the selected reuse threshold and $m_{\mathrm{CEP}} = 0.005$ its source-calibrated gate margin. The candidate is directly reused when $\hat{\mu}(x, u) > \eta_{\mathrm{CEP}} + m_{\mathrm{CEP}}$. It is probe-eligible when $|\hat{\mu}(x, u) - \eta_{\mathrm{CEP}}| \le m_{\mathrm{CEP}}$ and a complete paired probe fits the remaining budget. Otherwise, it abstains. Both gate quantities are fixed before target evaluation.

F.3 LOCAL-SUPPORT PREDICTION

CE-LOCAL uses compatible-context neighbors with $k \in \{8, 16, 32, 64\}$, distance bandwidth in $\{0.25, 0.5, 1\}$, and either uniform or inverse-distance weighting, giving 24 fitting settings. Distances use the same source-normalized features and categorical compatibility rules as the contract executor. Root-level bootstrap intervals use 500 resamples with an explicit support minimum. At target time, let $L_{\mathrm{local}}(x, u)$ denote the lower bootstrap bound. A compatible candidate is directly reused when it meets the minimum-support requirement and $L_{\mathrm{local}}(x, u)$ exceeds the source-selected reuse threshold. If it does not pass that threshold, it may use a complete paired probe when the remaining budget permits. Otherwise, the method abstains. Retrieval and resampling time are metered.

F.4 ADDITIONAL MEMORY-ADAPTATION SPECIFICATIONS

All four memory baselines use protocol-matched adaptations. They share the proposal model, legal intervention interface, source feedback, target resource caps, held-out evaluation, and fresh target-policy initialization with contracts.

A-MEM maps each source comparison to a linked structured note. The note stores the predecision context, intervention, comparator/horizon, observed effect, and evidence pointer. Its memory manager may add descriptors, links, and source-only revisions. SELFMEM instead optimizes a bounded menu of memory operations using source-only feedback. The allowed operations are add, merge, revise, reindex, retrieve, and deprecate. The resulting memory policy is frozen before target evaluation. MemRL retains episodic-memory reinforcement. MemSkill retains reusable memory-skill organization. For both methods, agent-facing records are replaced by trainer-evidence records. None receives

contract predicates or imported validity regions. All use the common legality, monitoring, retrieval, and probe interfaces at target time.

F.5 Hand-written gate and generic verifier

The hand-written gate is set before target evaluation. Reduce approach shaping when $r \geq 0.75$ and $h \leq 0.50$. Do not reduce it when $r < 0.40$. Lower the learning rate when normalized critic residual exceeds the source 90th percentile and development progress is nonpositive. Increase exploration when progress is nonpositive and action variability is below the source 25th percentile. Increase curriculum difficulty only when development success is at least 0.80. Rules apply only to compatible typed edits. The source tuning budget may select among the listed thresholds. The rules are not manually revised for individual target tasks.

The generic verifier receives the same candidate, counterexamples, development observations, legal probes, fallback, and static guards. It selects its confidence threshold on source calibration. It receives both positive and negative memories. Positive-only consolidation is evaluated separately as an ablation.

F.6 Offline compilation, permutation, and a bridge control

Offline compilation receives the complete common source evidence and the same final capacity but constructs contracts in one batch.

Permutation operates within intervention/adapter compatibility classes, swapping validity-region assignments among contracts with matched clause counts and similar discovery-support size. Intervention code, source-effect records, candidate retrieval order, token budget, and legal checks remain unchanged. Assignment seeds are set before target evaluation. We average 20 independent permutations for each state before treating a source lineage as one observation. The fixed-spend experiment and rank-matched audit compare alignment effects at matched rejection and probe budgets. Reruns and permutations use a diagnostic resource ledger.

The bridge control gives CE-U the learned contract validity regions while leaving its effect predictor and ranking intact.

G Lineage-Level Statistics and Uncertainty

G.1 Full definition of the primary score

At interaction boundary b_j, the trainer commits checkpoint $\hat{\imath}_j$ using only training and development observations. With task P, independent source lineage ℓ, target decision trajectory r, and policy seed s,

$$\bar{s}_{P\ell rj} = \frac{1}{S}\sum_{s=1}^{S} V^*(\pi_{\hat{\imath}_j, P\ell rs}), \tag{15}$$

$$c_{P\ell r} = \sum_{j=1}^{J_B} \frac{b_j - b_{j-1}}{B_P}\bar{s}_{P\ell rj}, \qquad \widehat{C_{\text{anytime}}} = \text{Avg}_{P,\ell,r}\, c_{P\ell r}. \tag{16}$$

The score lies in $[0, 1]$. Training, development episodes, failed branches, and probes share the interaction ledger. Protected evaluation is withheld from the trainer and separately metered. Three seeds sharing one proposal sequence are averaged before forming a trajectory observation. Table 10 summarizes uncertainty for the two primary contrasts.

G.2 Estimands and primary contrasts

For the fixed twelve-task suite, policy seeds are averaged within each decision trajectory, followed by equal averaging over trajectories within a lineage/task cell and tasks within a lineage. For methods

Table 10: Uncertainty for the two primary contract–CE-U contrasts.

Primary contrast	Mean	Simultaneous 98.33% interval	Positive observed lineages / 12
Contracts − CE-U, common evidence	0.050	[0.026, 0.074]	11
Contracts − CE-U, own source acquisition	0.052	[0.026, 0.078]	11

a, b, the lineage contrast is

$$d_\ell^{a-b} = \frac{1}{12}\sum_{P=1}^{12}\frac{1}{4}\sum_{r=1}^{4}\left(c_{P\ell r}^{a} - c_{P\ell r}^{b}\right).$$

Method pairing is defined by the randomization design. Under independent source acquisition, the comparison is between the resulting trainers within coupled exogenous blocks, without forcing their post-treatment source trajectories to match. For the two reported primary contrasts, we retain the original three-contrast Bonferroni critical value,

$$\bar{d} \pm t_{11,1-0.05/(2\cdot 3)}\frac{s_d}{\sqrt{12}},$$

so the displayed intervals are conservative after omitting the auxiliary feedback diagnostic. The intervals use the small-sample working model for independent lineage effects and are conditional on the fixed twelve-task suite.

G.3 Observed lineage-level contrasts

Table 11 reports the twelve lineage-level observations in a fixed display order across both contrasts. Estimation and inference use the paired lineage values in this same order.

Table 11: Lineage-level contract–CE-U contrasts under the two evaluation protocols.

Lineage ID	Common-evidence d_ℓ	Own-source d_ℓ
1	0.055	0.020
2	−0.010	0.064
3	0.016	−0.007
4	0.028	0.052
5	0.050	0.050
6	0.061	0.055
7	0.067	0.046
8	0.085	0.074
9	0.075	0.125
10	0.043	0.043
11	0.095	0.065
12	0.035	0.037
Mean	0.050	0.052
Sample standard deviation	0.029885	0.032

The simultaneous intervals from these lineage-level inputs are $[0.026, 0.074]$ and $[0.026, 0.078]$.

G.4 Trajectory schema and outage handling

Each trajectory records protocol role, source state, task/configuration, coupled randomization block, proposal/repair events, predecision features, gate outcome, branch costs, committed checkpoint, and held-out scores. Checkpoints are committed before held-out scores are revealed. The outage rule repeats the full coupled block under its original assignment, records spent resources, and does not select outcomes using the rerun result. Residual missing outcomes remain in the assigned denominator.

H DECISION AUDIT, RISK–COVERAGE COUNTS, AND LABELS

The independent audit contains 960 cases: $N_+ = 412$, $N_- = 384$, $N_0 = 63$, and $N_? = 101$. All methods use the same prescribed 12% probe allocation. Neutral and unresolved cases remain in coverage.

Table 12: Fixed-allocation decision audit.

Fixed-12% diagnostic	C_{anytime}	Benefit retained / 412	Harm blocked / 384	Total accepted / 960
Generic verifier	0.505	328	280	472
Hand-written gate	0.548	338	312	447
CE-POINT	0.570	348	328	439
CE-U	0.580	352	339	433
Training Contracts (Ours)	**0.632**	**367**	**359**	428
Matched predicate permutation	0.514	341	282	483

H.1 AUDIT POPULATION AND INDEPENDENT LABELS

The audit samples 80 predecision cases from each of 12 independent source-lineage panels, giving 960 cases. Sampling uses a method-independent target checkpoint generator, the same decision windows, and legal candidate classes before audit outcomes are observed. The counts $N_+ = 412$, $N_- = 384$, $N_0 = 63$, and $N_? = 101$ are observed labels, not outcome-based sampling quotas.

Each audit case sets the context, intervention, horizon, comparator, and decision before effect evaluation. Effects use eight independent paired continuation groups from the audit checkpoint. Each group uses the same policy-seed panel and 0.512 million interactions per branch. Their randomness is independent of development probes and unavailable to the trainer.

With practical-effect margin 0.02, a case is beneficial when its independent interval lies above 0.02 and harmful when it lies below -0.02. It is neutral when the interval lies within $[-0.02, 0.02]$ and unresolved otherwise. Count tables use this fixed margin. Accepted unresolved cases are treated as harms in the conservative sensitivity analysis.

H.2 COUNT DECOMPOSITION FOR THE FIXED-ALLOCATION TABLE

The decomposition in Table 13 uses $A_+, A_-, A_0, A_?$ for accepted beneficial, harmful, neutral, and unresolved counts, respectively. Coverage is A/N with $A = A_+ + A_- + A_0 + A_?$. Benefit retention is A_+/N_+. Harm blocking is $1 - A_-/N_-$. Identified harmful fraction among accepted is A_-/A, and the unresolved-case sensitivity upper bound is $(A_- + A_?)/A$.

Table 13: Accepted-case decomposition for the fixed-allocation audit.

Fixed-12% operating point	A_+	A_-	A_0	$A_?$	Total A
Generic verifier	328	104	21	19	472
Hand-written gate	338	72	21	16	447
CE-POINT	348	56	20	15	439
CE-U	352	45	20	16	433
Training Contracts (Ours)	**367**	**25**	21	15	428
Predicate permutation	341	102	21	19	483

H.3 SOURCE-CALIBRATED DEPLOYED POLICY VERSUS EQUAL-RANK DIAGNOSTIC

At 45% coverage, treating every accepted unresolved case as harmful gives harm fractions of 9.0% for Contracts and 14.1% for CE-U, compared with known-harm fractions of 5.8% and 11.3%. The deployed operating threshold is chosen from source calibration before the target audit. Its acceptance fraction is therefore an outcome of the policy. The equal-rank diagnostic instead scores all audit cases and accepts the highest-ranked K under a chosen capacity. K is set without using audit labels. At

high coverage, the diagnostic may include cases that the deployed policy would normally abstain from.

Contracts rank supported direct-reuse candidates by the matched source lower bound. Unresolved candidates are ranked by a probe-value score calibrated on source data. Candidates with evidence contradicting reuse are ranked last. Ranking within a tier uses predecision fields and a canonical tie break. CE-U uses its calibrated lower-effect score and its own unsupported-input tier. Permutation applies the same ranking construction after validity-region reassignment. Table 14 gives all eight evaluated coverages. The 100% endpoint necessarily erases selection differences. Each row contains $K = 960 \times$ coverage accepted cases, and the plotted curves interpolate only between these measured operating points.

Table 14: Risk–coverage counts across eight operating points.

Coverage	Method	K	A_+	A_-	A_0	$A_?$
10%	CE-U	96	85	2	5	4
10%	**Training Contracts** (Ours)	96	**90**	**0**	4	2
10%	Permutation	96	74	8	8	6
25%	CE-U	240	214	10	10	6
25%	**Training Contracts** (Ours)	240	**223**	**4**	7	6
25%	Permutation	240	181	35	14	10
35%	CE-U	336	290	21	17	8
35%	**Training Contracts** (Ours)	336	**304**	**10**	14	8
35%	Permutation	336	251	53	18	14
45%	CE-U	432	348	49	23	12
45%	**Training Contracts** (Ours)	432	**369**	**25**	24	14
45%	Permutation	432	313	80	19	20
55%	CE-U	528	378	98	29	23
55%	**Training Contracts** (Ours)	528	**399**	**65**	39	25
55%	Permutation	528	352	123	24	29
65%	CE-U	624	397	156	34	37
65%	**Training Contracts** (Ours)	624	**407**	**130**	44	43
65%	Permutation	624	382	175	28	39
80%	CE-U	768	410	241	52	65
80%	**Training Contracts** (Ours)	768	**412**	**219**	63	74
80%	Permutation	768	405	253	45	65
100%	CE-U	960	**412**	**384**	63	101
100%	**Training Contracts** (Ours)	960	**412**	**384**	63	101
100%	Permutation	960	**412**	**384**	63	101

At 45% coverage, retained-benefit fractions are $348/412 = 84.5\%$ for CE-U and $369/412 = 89.6\%$ for contracts. The corresponding known-harm counts are 49 and 25.

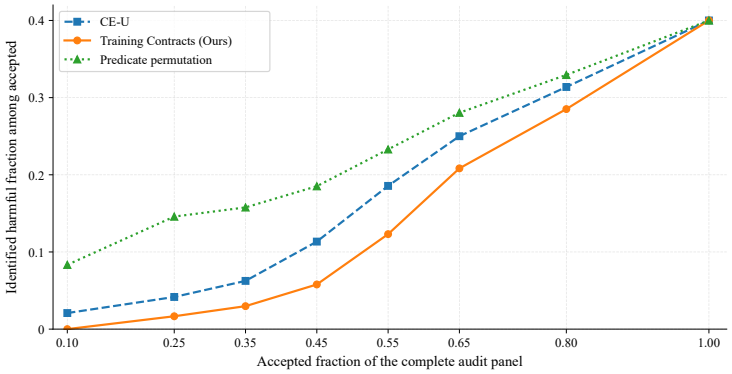


Figure 5: Known harmful interventions among accepted cases across eight matched coverage levels. Acceptance ranks use predecision scores, not audit labels. Unresolved cases remain in coverage but are not counted as known harms. Table 14 reports the full counts.

I Contract Example and Hidden-Context Failure

The worked example is the first contract in canonical intervention order with both positive and negative discovery evidence, selected before target effects are inspected.

I.1 Intervention and feature definitions

The edit changes only the auxiliary approach-shaping coefficient for the next training window:

```
# Auxiliary reward-coefficient edit. All other interfaces stay fixed.
assert cfg.reward.aux_approach_weight == 1.0
candidate_cfg = copy_of_current_training_config(cfg)
candidate_cfg.reward.aux_approach_weight = 0.25
# Keep controller, action map, observations, success, and termination
    unchanged.
```

The comparator retains coefficient 1.0. Both branches start from the same source checkpoint and receive identical continuation and development budgets. Features r and h follow Table 8. Target held-out scores and latent-shift labels are excluded from the predicate.

I.2 Discovery retains the counterexample

The initial source-discovery candidate reduces the auxiliary coefficient whenever it equals 1.0, without conditioning on training context. Discovery comparisons show benefit in high-reach/low-completion contexts but inconsistent benefit in the intermediate-reach region. Low-reach comparisons contradict the broad candidate and remain as counterevidence. The discovery objective narrows the validity region to $r \geq 0.80 \wedge h \leq 0.35$. Thresholds 0.80, 0.35, and 0.50 are serialized values from the discovery quantiles. The aggregate strata in Table 15 characterize the selected boundary, while exact reconstruction of quantile selection depends on the underlying feature observations.

Table 15: Discovery evidence for the approach-reward contract.

Discovery stratum	Root blocks	Mean effect	Role in revision
$r < 0.50$	32	-0.108	Retain as counterevidence
$0.50 \leq r < 0.80$	32	$+0.028$	Insufficient basis for direct reuse
$r \geq 0.80,\ h \leq 0.35$	32	$+0.181$	Candidate supported region
$r \geq 0.80,\ h > 0.35$	32	$+0.006$	Do not claim a useful effect

The supported and contradicted regions are frozen before confirmation outcomes are revealed. The thresholds remain frozen during target evaluation. The intermediate region remains unresolved and uses a bounded probe when affordable.

I.3 Independent confirmation

Table 16: Independent confirmation of the supported and contradicted regions.

Frozen confirmation region	Independent roots	Mean $\bar{d}$	Root SD s_d
Positive: $r \geq 0.80 \wedge h \leq 0.35$	24	$+0.172$	0.060
Negative: $r < 0.50$	24	-0.094	0.060

With the 64-test, two-direction allocation, both intervals use $t_{23,1-0.000390625}$. The positive lower bound is

$$0.172 - t_{23,1-0.000390625}\frac{0.060}{\sqrt{24}},$$

which exceeds 0.02. The negative upper bound

$$-0.094 + t_{23,1-0.000390625}\frac{0.060}{\sqrt{24}}$$

is below -0.02. The observed results satisfy the specified confirmation criterion. Discovery and confirmation share the comparator, horizon, feature scale, and tested region, while using independent roots.

I.4 Target execution, rollback, and the unobserved factor

Table 17: Illustrative target decisions for the approach-reward contract.

Predecision context	r	h	Decision	Independent paired effect
Acquisition not established	0.30	0.10	Abstain	-0.100
Approach reliable, completion weak	0.89	0.22	Reuse	$+0.180$
Boundary region. Validity unresolved	0.65	0.28	Probe	$+0.060$
Same visible features. Hidden actuator delay	0.89	0.22	Reuse, then rollback	-0.080

The fourth row of Table 17 adds an unobserved two-control-step actuator delay after checkpoint selection. With matched visible r, h, reuse becomes harmful and the common development monitor rolls back to the preceding target-local checkpoint.

J Portability and Controlled Mechanism Comparisons

J.1 Backbone, learner, and dynamics changes

The portability panels change one factor at a time. The backbone panel swaps the proposal model. The learner panel uses the SAC adapter from Appendix C. The dynamics panel shifts mass multipliers to $[1.5, 2.0]$ and friction to $[0.25, 0.45]$. Unsupported or incompatible interventions are rejected. The Meta-World adapter keeps only diagnostics and interventions with compatible semantics. Missing fields remain unknown and lead to probing or abstention.

Table 3 reports the backbone, learner, and dynamics results in the main text. The primary-backbone/PPO row repeats Table 1. The other rows are separate panels and are not paired across rows.

For Meta-World, the source trainer is evaluated on the official ML10/ML45 held-out test suites without Meta-World source learning or tuning. Only semantically compatible interventions and diagnostics are exposed through the adapter. Values are mean $\pm$ standard deviation across independent source-state constructions.

Table 18: Cross-benchmark transfer to Meta-World ML10 and ML45 test tasks. Avg is the arithmetic mean of the two suite means.

Method	ML10 test	ML45 test	**Avg**
CE-LOCAL: support-aware local predictor	0.475 ± 0.016	0.438 ± 0.019	0.457
CE-U: tuned uncertainty-aware predictor	0.493 ± 0.017	0.459 ± 0.011	0.476
Eureka (Ma et al., 2024a)	0.468 ± 0.037	0.424 ± 0.013	0.446
Text2Reward (Xie et al., 2024)	0.509 ± 0.021	0.460 ± 0.018	0.485
A-MEM(Xu et al., 2025)	0.441 ± 0.017	0.399 ± 0.020	0.420
SELFMEM (Yang et al., 2026)	0.427 ± 0.018	0.386 ± 0.021	0.407
MemRL (Zhang et al., 2026b)	0.518 ± 0.009	0.484 ± 0.013	0.501
MemSkill (Zhang et al., 2026a)	0.438 ± 0.032	0.416 ± 0.027	0.427
Training Contracts (Ours)	$\mathbf{0.532 \pm 0.014}$	$\mathbf{0.503 \pm 0.015}$	**0.518**

Mechanism strata are defined without target success. Dense support requires at least 48 compatible source-root comparisons and no source-observed sign reversal. The sign-reversal stratum requires positive and negative source effects in distinct observed strata. New combinations remain inside the allowed context ranges but were not jointly present in discovery. The latent-shift stratum changes a factor that is intentionally excluded from the registry.

K PHYSICAL SO-101 PROTOCOL

The physical evaluation reuses the same source trainer as the Meta-World panel. That trainer is produced by the primary ManiSkill/Panda source pipeline before any SO-101 training. Neither Meta-World nor SO-101 data enter source learning, contract discovery, confirmation, or tuning. The resulting trainer is evaluated independently on Cube Pushing and Cube Pick-and-Place as two held-out physical target tasks.

Each SO-101 target task starts from a fresh PPO learner. It uses the same actor/critic architecture and optimization defaults as the primary evaluation (Appendix C). Policy parameters, optimizer state, observation-normalization statistics, checkpoints, and target-local history are reinitialized for every run. Nothing learned on one physical target is transferred to the other. As in Meta-World, only trainer-side intervention experience is transferred. The robot-side adapter presents state-based proprioceptive and task-state observations through the same predecision feature interface used by the contract executor. Actions are bounded end-effector delta-pose commands plus a gripper command. They are clipped to a fixed Cartesian workspace. Episodes use the same 200-step decision horizon as the primary manipulation suite.

The SO-101 adapter keeps only diagnostics and intervention types whose meaning is preserved across embodiments. Reward-coefficient edits over auxiliary terms are eligible only when those quantities retain the same meaning on the physical task. The same rule applies to PPO learning-rate or entropy/exploration edits and to source-defined curriculum or reset-difficulty changes. Diagnostics are limited to compatible quantities such as development success, progress, budget fraction, action variability or saturation, and learner-side residual statistics when available. Joint-specific controller edits, simulator dynamics edits, and interventions that cannot be mapped without changing meaning are unsupported and are not reused.

Compared methods receive the same robot-training budget and four target decision windows. Paired probes use at most the primary 20% probe share. Unused probe budget returns to incumbent training. For each method and task, we conduct 10 independent target-policy training runs from scratch. For every run, we reinitialize policy parameters, optimizer state, observation-normalization statistics, checkpoints, and training randomness. Each trained policy is evaluated on six held-out trials, yielding 60 evaluation trials per method and task. Final success is aggregated across these 60 trials. We compute C_{anytime} for each training run and then average it across the 10 independent runs. Training interactions, probe branches, failed intervention branches, and recovery motions are charged to the target budget. Protected held-out evaluations are used only for reporting and never for training or contract decisions.

Both tasks use the same cubic object and tabletop workspace across methods. Resets randomize the cube and goal locations within predefined reachable regions, with the same held-out evaluation protocol for every method. Cube Pushing requires moving the cube into the marked planar goal region without lifting it. Cube Pick-and-Place requires grasping, relocating, releasing, and leaving the cube stably inside the goal region. A held-out evaluation is successful only when the task-specific goal predicate holds at the end of the episode. Evaluation resets are fixed before method comparison and are not reused for development decisions.

Robot execution is constrained to the tabletop workspace with clipped Cartesian motion and gripper commands. A hardware emergency stop and software workspace/collision guards remain active throughout training and evaluation. Operator intervention is allowed only for safety or hardware faults. A safety interruption caused by the executed policy counts as a failed trial and remains charged. A verified external hardware fault triggers a full rerun of the affected trial. Policy-caused safety interruptions remain failed trials.