TEMPORAL SELF-IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon policies trained with reinforcement learning can still achieve high return through inefficient interactions, while rare efficient behaviors discovered during training may be forgotten. We argue that temporal efficiency itself provides a source of self-supervision for reinforcement learning. We introduce **Temporal Self-Imitation Learning (TSIL)**, a reinforcement learning framework that mines temporally efficient successful trajectories generated during learning and converts them into reusable supervision for future policy improvement. TSIL progressively refines learning using configuration-conditioned adaptive temporal targets derived from fast successful trajectories, while preserving and replaying efficient behaviors through efficiency-weighted self-imitation learning. Across 30 long-horizon tasks spanning robot manipulation and interactive navigation, TSIL consistently improves learning efficiency, behavioral efficiency, revisitation of fast successful behaviors, and robustness to unstable training conditions.

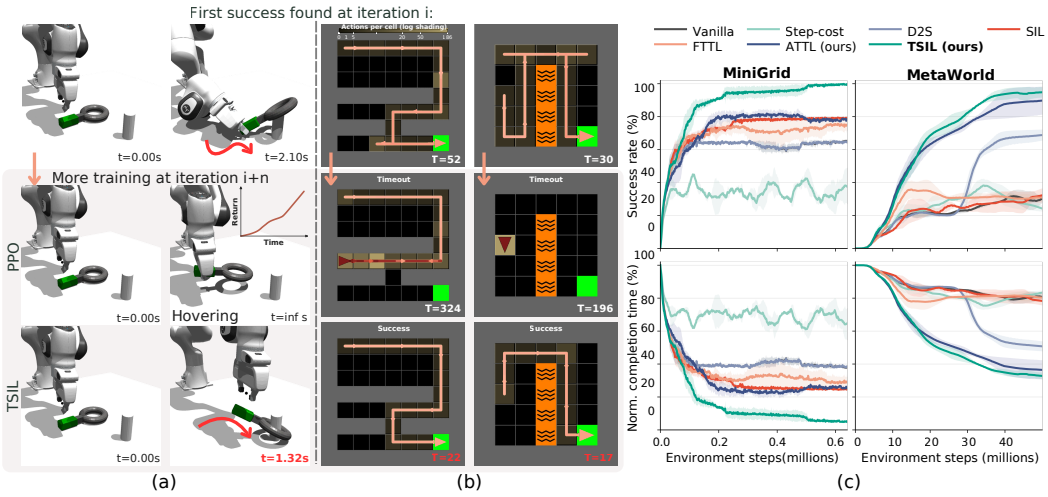

Fig. 1: **TSIL focuses on three challenges in long-horizon RL. (a) Distraction:** In a dense-reward environment, the policy may exploit shaping rewards and drift away from previously discovered successful behavior. TSIL instead regularizes learning toward a temporal target derived from the previous fastest success. **(b) Forgetting:** In a sparse-reward environment, the policy may fail to retain successful behavior after discovering it. TSIL helps the policy revisit actions from previous fast successes. **(c) Efficiency:** TSIL continually refines fast successful behavior, improving both training and behavioral efficiency for task completion.

1 INTRODUCTION

Motor learning in humans and animals progressively refines memorized behavior toward more efficient and reliable movement. Through repeated interaction and reinforcement, biological systems learn to suppress unnecessary motion, reduce behavioral variability, and discover increasingly direct strategies for achieving a goal (Krakauer et al., 2019; Vassiliadis et al., 2021). Skilled behavior therefore emerges not only from eventual task success, but from refining success into efficient and reproducible interaction patterns.

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.

In contrast, long-horizon reinforcement learning can struggle to turn a discovered success into reliable future behavior. Reward objectives may imperfectly reflect efficient task completion, so useful behavior may not be reinforced by subsequent updates even when it remains represented in the data. Continued policy updates also shift the distribution of collected trajectories, leading rare successes to disappear from later training. Together, these issues can prevent a discovered success from becoming an efficient and reliable behavior.

Specifically, one problem with dense rewards is **distraction**. Long-horizon manipulation policies are commonly trained with shaping rewards alongside task-success rewards (Gu et al., 2017; Schulman et al., 2017; Yu et al., 2020). A robot may exploit high-reward intermediate states, remain stuck near partially solved configurations, or take much longer manipulation paths before eventually completing the task. Such behavior can achieve high return without solving the task efficiently, as shown in Figure 1(a).

With sparse rewards, another problem is **forgetting**. Successful episodes may be rare, and subsequent policy updates can move the agent away from the successful behavior. Figure 1(b) illustrates this failure mode. In this case, the policy needs to rediscover the success through new interactions. The policy may then repeatedly forget and relearn the same successful behavior, leading to inefficient learning and unstable performance.

Existing approaches address parts of this problem through manually designed temporal preferences or experience replay. For distraction, per-step penalties can favor shorter trajectories (Ng et al., 1999), while dense-to-sparse reward schedules (Luo et al., 2020) and time-limit handling (Pardo et al., 2018) alter the temporal structure of learning. These methods rely on hand-designed penalties, schedules, or horizons that require task-specific tuning. Weak temporal signals may have little effect, whereas strong ones may hinder exploration. They do not use the temporal structure of the agent's own successful behavior to set and continually refine targets for subsequent learning. For forgetting, standard self-imitation learning reuses high-return experience (Oh et al., 2018), which can help preserve success, but return alone may not distinguish fast successes from slower ones. This can focus the policy on irrelevant interactions instead of encouraging more efficient solutions.

Our central hypothesis is that temporally efficient successful behavior discovered during learning contains reusable self-supervisory signals for future policy improvement. Once a policy discovers that a task can be solved efficiently, subsequent learning should bias toward behaviors that match or exceed this efficiency, allowing temporal targets and efficient behaviors to be discovered and refined by the policy itself.

We present **Temporal Self-Imitation Learning (TSIL)**, a reinforcement learning framework that mines temporal efficiency from self-generated successful trajectories and converts it into reusable supervision. TSIL uses fast successful trajectories in two complementary ways. First, the completion time of a successful trajectory becomes an adaptive temporal target that conditions future observations and rewards. As faster solutions emerge, this target tightens and encourages increasingly efficient behavior. Second, efficient successful trajectories are preserved in a replay buffer and reused through efficiency-weighted self-imitation, helping the policy revisit behaviors that may otherwise disappear under policy distribution shift.

We evaluate TSIL on 30 long-horizon tasks with dense and sparse rewards, state-based or image-based observations, and continuous or discrete control: 15 robot manipulation tasks spanning articulated-object interaction, insertion, tool use, transport, and contact-rich manipulation, and 15 MiniGrid tasks including navigation and object interaction challenges. Across these settings, TSIL consistently improves learning efficiency, behavioral efficiency, successful behavior revisitation, and robustness under unstable training conditions.

In summary, our contributions are:

- We identify temporal efficiency as a self-supervisory signal for long-horizon reinforcement learning, and show that reward alone can fail to distinguish efficient successful behavior from slower or reward-distracted success.
- We introduce Temporal Self-Imitation Learning, which combines adaptive temporal targets with efficiency-weighted self-imitation of successful trajectories.
- We demonstrate consistent improvements in learning efficiency, behavioral efficiency, successful behavior revisitation, and robustness across 30 long-horizon tasks spanning robot manipulation and MiniGrid.

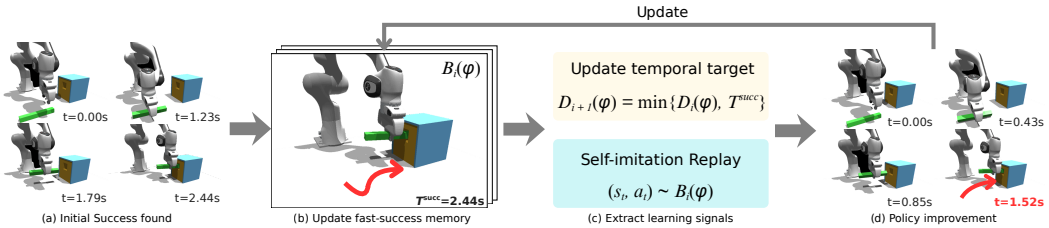

Fig. 2: **TSIL training loop.** (a) The policy discovers a successful trajectory during early interactions. (b) The trajectory updates the policy's fast-success memory. (c) Its completion time updates the adaptive temporal target, while its transitions provide self-imitation supervision. (d) The updated policy is encouraged to complete the task more efficiently, and faster successful trajectories feed back into the memory.

2 Related Work

Temporal reinforcement learning. Temporal reinforcement learning studies how time enters the decision process, from finite-horizon state augmentation and time-limit handling (Pardo et al., 2018) to continuous-time or discretization-robust control (Doya, 2000; Tallec et al., 2019; Yildiz et al., 2021; Ramstedt & Pal, 2019) and temporal abstraction through options (Sutton et al., 1999), action repetition (Lakshminarayanan et al., 2017; Sharma et al., 2017), or learned action timing (Biedenkapp et al., 2021). Recent time-aware RL methods further address irregular decision intervals (Kim & Chi, 2021; 2023), time-step-conditioned world models (Nhu et al., 2025), and timing-aware driving or robot navigation (Li et al., 2024; Chen et al., 2023). Adjacent safe offline RL also conditions trajectory planning on dynamically specified deployment budgets, though these budgets denote cost constraints rather than task-time variables (Lin et al., 2023). Closest to our setting, time-aware policy learning for robot control conditions a trained manipulation policy on remaining time and a time ratio, enabling inference-time tempo adaptation (Jia & Chen, 2026). In contrast, TSIL uses time-awareness during training: the policy's own fast successes set configuration-conditioned adaptive temporal targets that reshape rewards, observations, and self-imitation priorities.

Self-imitation learning. Self-imitation learning uses an agent's own experience as supervision. The original SIL objective imitates high-return trajectories through advantage filtering (Oh et al., 2018), while later variants extend this idea with episodic hindsight replay (Dai et al., 2020), sparse or delayed robot rewards (Chen & Lin, 2021), and planning-based robot motion learning (Luo et al., 2021; Luo & Schomaker, 2023). In robot learning, successful experience has also been reused through sparse-reward relabeling (Bujalance & Moutarde, 2023), task reduction (Li et al., 2021), and goal-conditioned self-imitation (Li et al., 2022; 2023). Recent self-improving robot systems further turn autonomous practice into supervision for manipulation and foundation-model post-training (Sharma et al., 2023; Bousmalis et al., 2023; Seyed Ghasemipour et al., 2025). TSIL differs by making replay selection temporal: it prioritizes fast successful trajectories and relabels them under the current adaptive temporal target.

3 Temporal Self-Imitation Learning

The central idea of Temporal Self-Imitation Learning (TSIL) is to treat temporally efficient successful behavior during learning as reusable self-supervision for future policy improvement. Long-horizon reinforcement learning presents two coupled challenges. Policies can over-exploit dense rewards through slow or reward-distracted behavior rather than completing the task efficiently. Meanwhile, the rare fast successes that reveal better behavior may disappear during later on-policy updates due to distribution shift or unstable optimization. TSIL addresses these challenges by mining temporal efficiency from successful trajectories and feeding it back into future learning. The completion time of efficient successful trajectories becomes a progressively refined temporal target, while the trajectories themselves are replayed through efficiency-weighted self-imitation. Once efficient behavior emerges, future optimization increasingly biases toward reproducing and improving upon it.

3.1 TEMPORAL TARGET CONDITIONING

Robot policies are commonly trained with an infinite-horizon discounted objective, $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})]$, using truncated rollouts and bootstrapping. We use an infinite-horizon formulation because the number of steps needed to complete a robot task is generally unknown. Task difficulty varies across configurations, and disturbances may require additional interaction for recovery. Although discounting weakly favors shorter trajectories, tuning it is a blunt temporal preference: small discount factors can suppress delayed success signals, while large ones only mildly favor faster completion. Moreover, it does not explicitly distinguish efficient successful behavior from slow or reward-distracted behavior. In long-horizon tasks, policies can still accumulate substantial return through prolonged recovery sequences or repeated exploitation of dense shaping rewards before eventually succeeding.

Instead of manually engineering temporal objectives, TSIL derives temporal supervision directly from successful behavior discovered during learning. Let ϕ denote a task configuration, such as initial state, goal, and environment configuration. Let N be the episode horizon, Δt the control interval, and $T^{\max} = N\Delta t$ the maximum rollout duration.

A central design choice is that temporal targets are configuration-conditioned rather than global. The same task can have very different feasible completion times depending on ϕ, so a target discovered for an easy configuration should not be imposed on a harder one. For each training configuration, TSIL maintains an adaptive temporal target $D(\phi) \leq T^{\max}$, representing the fastest successful completion time discovered so far under comparable initial conditions. The policy receives the original observation together with elapsed interaction time and the active target, $\tilde{o}_t = (o_t, T_t^{\text{used}}, D(\phi))$, where $T_0^{\text{used}} = 0$ and $T_{t+1}^{\text{used}} = T_t^{\text{used}} + \Delta t$.

Importantly, the temporal target is not a hard environment constraint. Rollouts may continue beyond $D(\phi)$ until task completion or rollout truncation at $T^{\max}$. Instead, the temporal target acts as adaptive supervision that progressively biases learning toward increasingly efficient behavior. The resulting objective is $J(\pi; D) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}, T_t^{\text{used}}, D(\phi))]$.

3.2 ADAPTIVE TEMPORAL SUPERVISION

A successful trajectory reveals that a specific configuration can be solved within a particular amount of interaction time. Let D_i denote the temporal target function at training iteration i, initialized as $D_1(\phi) = T^{\max}$. For a successful trajectory τ, let K_τ be its completion step count and $T^{\text{succ}}(\tau) = K_\tau \Delta t$. After each iteration, TSIL updates the target using the fastest successful trajectory discovered so far:

$$D_{i+1}(\phi) = \min\left(D_i(\phi), \min_{\substack{\tau \in \mathcal{T}_i(\phi) \\ \mathbb{1}_{\text{succ}}(\tau)=1}} T^{\text{succ}}(\tau) \right), \tag{1}$$

where the inner minimum is ignored if no successful trajectory for ϕ appears in $\mathcal{T}_i(\phi)$. Thus, temporal targets progressively tighten as the policy discovers more efficient successful behavior.

We further use the temporal target to shape the terminal success reward. For a trajectory terminating after K steps with elapsed interaction time $T^{\text{used}} = K\Delta t$, the target-conditioned success reward is

$$r_{K-1}^{\text{succ}} = r_{\text{succ}}\, \mathbb{1}_{\text{succ}} \left[1 + \min\left(\frac{D_i(\phi)}{T^{\text{used}}}, 1 \right) \right]. \tag{2}$$

This reward preserves the task-completion incentive while adding a temporal bonus relative to the current target. Trajectories that finish before the target receive maximal temporal bonus, while slower trajectories receive less as T^{used} grows. With task rewards, the final reward is $r_t = r_t^{\text{task}} + r_t^{\text{succ}}$.

3.3 EFFICIENCY-WEIGHTED SELF-IMITATION

Adaptive temporal supervision biases the current rollout toward efficient behavior, but efficient successful trajectories may still disappear during later policy updates. TSIL therefore stores temporally efficient successful trajectories in an auxiliary replay buffer. Unlike standard self-imitation learning,

which prioritizes high-return trajectories (Oh et al., 2018), TSIL prioritizes fast successful trajectories because high return can still contain slow or reward-distracted behavior.

For each configuration ϕ, TSIL maintains a replay buffer $\mathcal{B}_i(\phi)$ of capacity k that retains the fastest successful trajectories discovered so far. Before replay, the current replay buffer is relabeled using the current target D_i, yielding $\hat{\mathcal{B}}_i(\phi)$. For each trajectory τ collected under configuration ϕ^τ, the observation becomes $\hat{o}_t^\tau = (o_t^\tau, T_t^{\text{used},\tau}, D_i(\phi^\tau))$, the reward becomes $\hat{r}_t^\tau = r_t^{\text{task},\tau} + \mathbb{1}(t = K_\tau - 1)\hat{r}_{K_\tau-1}^{\text{succ},\tau}$, and the relabeled return is $\hat{G}_t^\tau = \sum_{j=t}^{|\tau|-1} \gamma^{j-t}\hat{r}_j^\tau$, where $|\tau|$ is the stored trajectory length.

Not every transition in a successful trajectory should be imitated. Following self-imitation learning, TSIL uses a positive return-value gap, $A_t^+(\tau) = [\hat{G}_t^\tau - V_\psi(\hat{o}_t^\tau)]_+$, as a soft gate so replay pressure vanishes once the critic judges the current policy to match the stored behavior. To further prioritize temporally efficient success, trajectories are weighted by $w_i(\tau) = 1 + \mathbb{1}_{\text{succ}}(\tau)\min(D_i(\phi^\tau)/T^{\text{succ}}(\tau), 1)$.

The full objective is $\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{SI}}\mathcal{L}_{\text{SI}}$, where the auxiliary self-imitation loss is

$$\mathcal{L}_{\text{SI}} = \mathbb{E}_{(\tau,t)\sim\hat{\mathcal{B}}_i}\left[-w_i(\tau)A_t^+(\tau)\log\pi_\theta(a_t^\tau \mid \hat{o}_t^\tau) + \frac{\lambda_{\text{V,SI}}}{2}\left(A_t^+(\tau)\right)^2\right]. \tag{3}$$

3.4 Implementation details

Algorithm 1 in Appendix A summarizes the complete TSIL training procedure. TSIL was implemented on top of PPO using a standard actor-critic architecture. PPO itself is unchanged except for temporal target conditioning, target-conditioned rewards, and the auxiliary efficiency-weighted self-imitation objective. Training uses parallel simulation environments, each maintaining its own task configuration, temporal target, and replay buffer. To balance replay across environments and prevent easy configurations from dominating it, each configuration stores the top-k ($k = 5$) fastest successful trajectories. During very early training, when successful trajectories are scarce, the replay buffer is supplemented with high-return trajectories to stabilize replay. Appendix A describes fallback replay and bootstrapped returns.

Within iteration i, rollout collection, replay relabeling, and policy optimization all use D_i. The updated target D_{i+1} is committed after optimization and applied at the next episode reset, keeping the rollout observations, rewards, and stored old-policy log probabilities consistent during the optimization. See Appendix A for the update schedule.

Replay sampling. TSIL filters transitions using the positive return-value gap and samples up to n positive transitions per trajectory without replacement, with probability proportional to the gap. The replay set is constructed once per PPO update, contains at most knN_{env} transitions, and is reused across E PPO epochs. Sampling details are provided in Appendix A.2.

Evaluation temporal target inference. For an unseen MiniGrid evaluation configuration, TSIL sets the temporal target to the mean target of up to five nearest successful (5-NN) training configurations from the same task. If no successful training configuration is available, TSIL uses the episode horizon. The inferred target remains fixed during evaluation. Configuration representations and distance measures are specified in Appendix A.2.

4 Experiments

We evaluate TSIL by asking three questions: **(1) Effectiveness:** does TSIL improve performance on long-horizon reinforcement learning tasks? **(2) Efficiency:** does TSIL improve both learning efficiency and behavioral efficiency? **(3) Robustness:** does TSIL improve robustness under unstable training conditions?

Tasks and training settings. We evaluate TSIL on 30 long-horizon tasks from two complementary domains. The 15 MiniGrid tasks (Chevalier-Boisvert et al., 2023) use partial image observations and discrete control with sparse rewards, spanning navigation and object-interaction challenges. The 15 Meta-World manipulation tasks (Yu et al., 2020), implemented in Isaac Gym through MT-Bench (Joshi et al., 2025), use state observations and continuous robot control with dense shaping

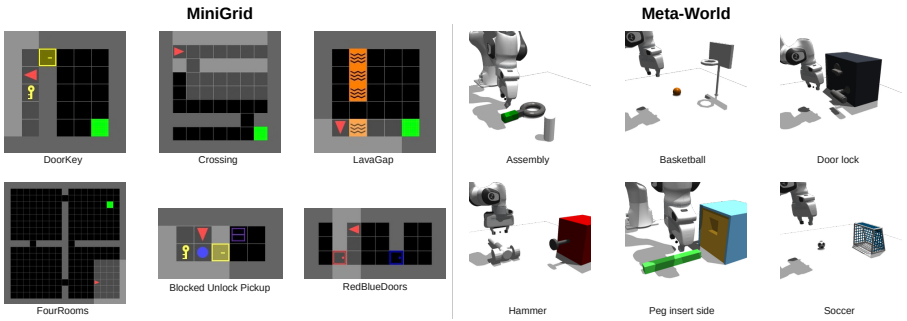

Fig. 3: **Representative tasks.** Left: MiniGrid tasks cover navigation and object interaction with sparse rewards, image observations, and discrete control. Right: Meta-World tasks cover assembly, tool use, articulated object interaction, and dynamic manipulation with dense rewards, state observations, and continuous control.

rewards. Each task is trained independently under its environment-specific horizon and interaction budget. For robustness tests, we apply policy-gradient noise and increased PPO clip ratios to both benchmarks, plus dense-reward dropout to Meta-World because MiniGrid rewards are already sparse. Implementation and evaluation details are provided in Appendix B.

Baselines. All methods use PPO (Schulman et al., 2017) as the base optimizer. We compare against (1) Vanilla PPO, (2) Step-cost (Chevalier-Boisvert et al., 2023), which subtracts the per-step cost $c(\phi) = 0.1 r_{\text{succ}}/T^{\max}(\phi)$, (3) dense-to-sparse reward scheduling (D2S) (Luo et al., 2020), which removes dense rewards at 50% of the training budget, (4) standard self-imitation learning (SIL) (Oh et al., 2018), (5) fixed temporal target learning (FTTL), which observes elapsed time and a fixed target equal to the episode horizon, and (6) adaptive temporal target learning without fast-success replay (ATTL). For fair comparison, SIL matches TSIL's replay budget, transition sampler, and optimization schedule, but stores the top-k ($k = 5$) highest-return trajectories per configuration instead of fast successes. It also disables adaptive temporal-target conditioning (elapsed-time and target observations) and temporal-efficiency weighting. All methods use the same base success-reward coefficient r_{succ} within each benchmark. Baseline settings and hyperparameters are listed in Appendix B.3.

Metrics. We report five primary metrics. (1) Training success rate is the best rolling success rate within the final 10% of training. (2) Area under the success rate curve (AUC) summarizes learning performance over the full interaction budget. (3) Successful episode count is the total number of successful episodes completed during training. (4) Normalized completion time is $T^{\text{succ}}/T^{\max}$ on success and 1 on failure, with lower values indicating more reliable and efficient completion. This is evaluated at the checkpoint with the best training success rate. (5) Evaluation success rate measures success over 2,000 evaluation episodes per task per seed, using the checkpoint with the highest rolling training success rate within the final 10% of training. This tests generalization to unseen configurations, including inference of their temporal targets. To study how adaptive temporal targets and efficiency-weighted replay alter learning, we additionally analyze positive-advantage mass, gradient update directions, and an efficiency-weighting ablation. Each method is evaluated with three random seeds per task. More details are provided in Appendix B.4.

4.1 TSIL IMPROVES EFFECTIVENESS, LEARNING AND BEHAVIORAL EFFICIENCY

As shown in Fig. 1(c) and Fig. 4, TSIL achieved the strongest overall performance across effectiveness, learning efficiency, and behavioral efficiency in both Meta-World and MiniGrid. In particular, TSIL achieved the highest training success rate and AUC while also obtaining the lowest normalized completion time, the largest number of successful training episodes, and the highest evaluation success rate. These results support our key hypothesis that temporally efficient successful trajectories provide self-supervision for policy improvement. Appendix C provides success-rate and completion-time training curves for all 15 tasks in each benchmark.

The progression across baselines further clarifies the contribution of each component. Reward-engineering baselines (Step-cost and D2S) improved some individual metrics, but remained sub-

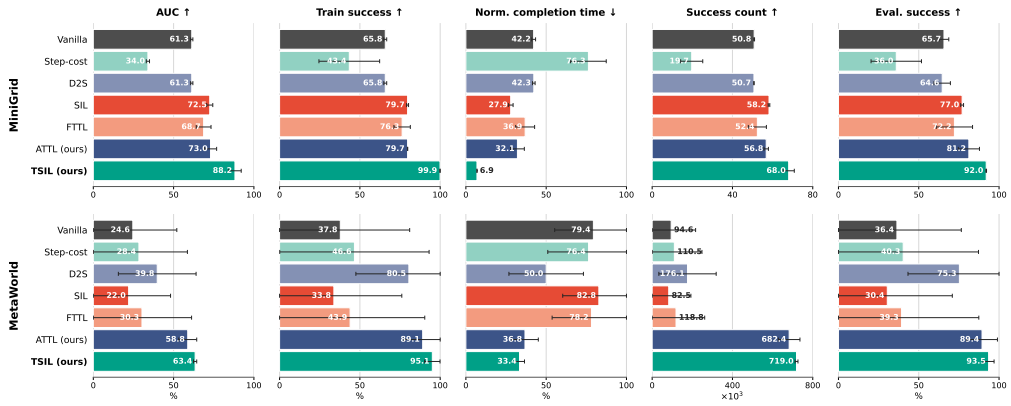

Fig. 4: **Main results under standard training**. TSIL performs best overall in task success, learning efficiency, and behavioral efficiency. Values are mean ± standard deviation across tasks and seeds.

stantially weaker than temporal-target-based methods in overall success and efficiency. FTTL improved training success over Vanilla in both domains, showing that temporal conditioning alone was beneficial, yet its fixed temporal target did not consistently encourage successful and efficient task completion. Introducing adaptive temporal targets (ATTL) substantially improved learning and behavioral efficiency by progressively tightening supervision toward the shorter temporal targets discovered during learning. This reduced distraction from slow or suboptimal behavior and improved both training and evaluation success rates. SIL improved success and learning efficiency in MiniGrid but remained weaker on Meta-World. This suggests that broader high-return experience can help in the sparse-reward environment, but high-return replay can also direct learning away from efficient task completion in dense-reward settings. In contrast, full TSIL consistently achieved the best overall performance in task success, learning efficiency, and behavioral efficiency, demonstrating that replaying temporally efficient successful trajectories provided substantially more effective supervision than replaying generic high-return experience. The high evaluation success rate also demonstrates strong generalization of the estimated temporal targets to new configurations. We also observed fewer unnecessary motions, parallel execution of motions, and more effective use of momentum within the controller's joint-velocity limits in Meta-World tasks. See the supplementary video for examples.

4.2 TSIL IMPROVES ROBUSTNESS UNDER UNSTABLE TRAINING

Long-horizon reinforcement learning is often highly sensitive to optimization instability. In practice, unreliable policy gradients and imperfect reward signals can cause policies to drift away from previously discovered efficient successful behaviors. Since TSIL explicitly preserves and replays temporally efficient successful trajectories, we hypothesize that it should also improve robustness under unstable training conditions by anchoring optimization to efficient successful behavior even when the current on-policy signal becomes unreliable. To evaluate this, we introduced three training disturbances for Meta-World and the two applicable disturbances for MiniGrid as discussed above.

Figure 5 reports the same five metrics for each disturbance class (settings in Appendix B.2). ATTL shows large performance drops across all disturbances. Without fast-success replay, unstable policy updates can rapidly move the policy away from previously discovered successful behaviors and lead to policy collapse. When dense reward supervision is unstable, ATTL may forget previously learned behavior and repeatedly rediscover it, leading to degraded performance and slower learning. TSIL consistently achieved the strongest overall performance, indicating that replaying efficient successful trajectories provided an effective optimization anchor against policy drift, behavior forgetting, and unreliable reward signals.

The disturbances also distinguish generic high-return replay from temporally efficient replay. SIL maintained competitive performance on MiniGrid and slightly exceeded ATTL on several metrics, but performed even worse in Meta-World. This comparison exposes a limitation of high-return-based replay: in dense-reward tasks, high return may correspond to slow, failed, or reward-distracted behavior and becomes even less reliable under reward dropout. Replay can mitigate forgetting, but

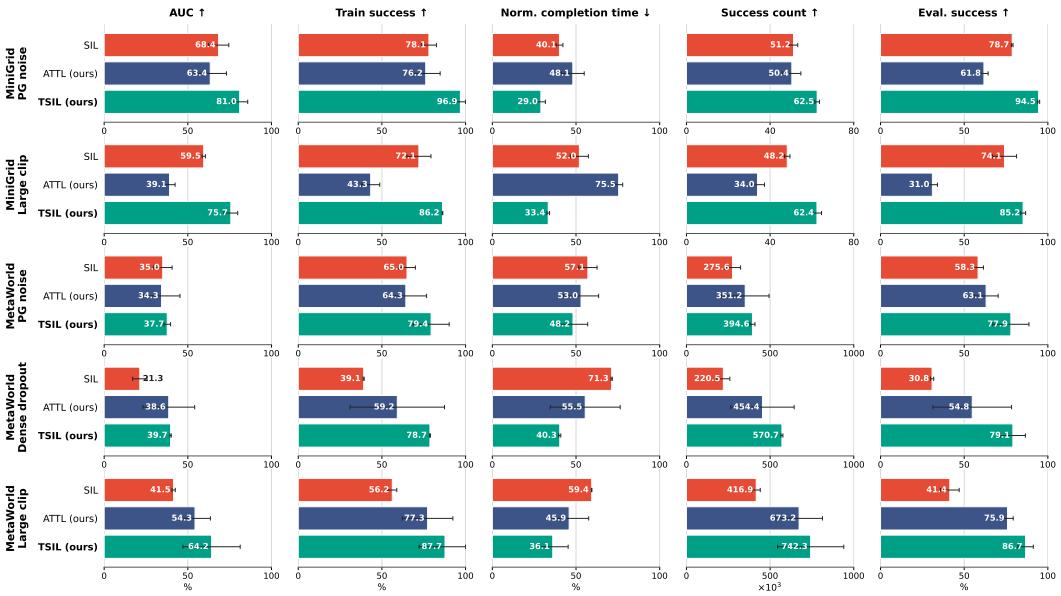

Fig. 5: **Main results under disturbed training.** ATTL degrades under disturbances. SIL improves MiniGrid but remains weaker in Meta-World. TSIL performs best overall, especially under severe policy drift and unreliable dense rewards. Values are mean ± standard deviation across disturbance levels, tasks, and seeds.

its selection criterion determines whether it preserves behavior that supports efficient task completion. In contrast, TSIL selects replay trajectories by successful completion and temporal efficiency. TSIL's clearer gains across all Meta-World disturbances indicate that fast-success replay anchors learning to efficient and reliable behavior, providing both effective supervision and stabilizing memory signals under unreliable rewards and unstable optimization.

4.3 Analysis

Why adaptive temporal targets help. We next analyzed how adaptive temporal targets altered PPO optimization before replay was added using Meta-World tasks. For each episode, we aggregated the positive-advantage mass used by PPO and grouped episodes by normalized completion time and dense task return. In Fig. 6, cell color indicates empirical success rate, while the overlaid color gradient indicates the fraction of positive-advantage mass assigned to that region. The summary bars further aggregate this update signal into four categories: success, fast-success, slow-failure, and reward-distracted behavior. Definitions and additional tasks: Appendices D.1–D.2.

As shown in Fig. 6, adaptive temporal targets rapidly concen-

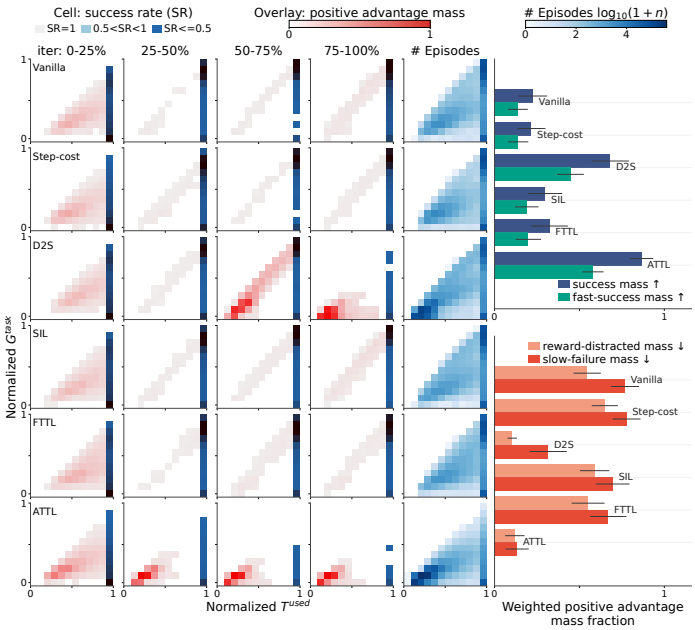

Fig. 6: **Adaptive temporal targets redirect PPO optimization.** Adaptive temporal targets rapidly concentrate positive PPO update signal on fast successful trajectories while suppressing reward-distracted and slow-failure behavior. Left: one representative Peg Insertion Side learning signal map. Right: aggregated results over all 15 tasks.

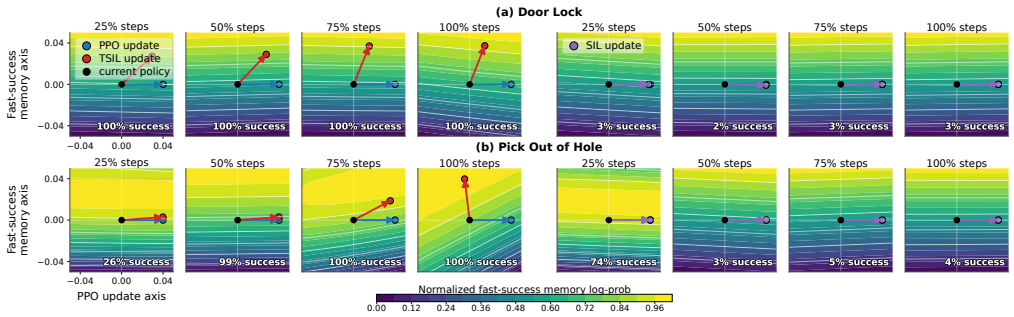

Fig. 7: **TSIL redirects updates toward fast-success behavior.** Across four training stages, TSIL aligns updates more strongly than SIL with high fast-success-memory likelihood. In (a), this promotes rapid exploitation of early successes. In (b), it preserves successful behavior and reduces later forgetting.

trated PPO update signal on fast successful trajectories while assigning little optimization pressure to reward-distracted or slow-failure behavior. Fixed temporal targets improved time awareness but could not distinguish achievable efficient behavior from generic timing constraints, allowing failed high-return trajectories to continue receiving positive updates. Vanilla, Step-cost, and SIL were substantially more vulnerable to reward distraction. D2S partially mitigated this effect by removing dense rewards at the cost of losing shaped guidance and wasting much of the training budget before switching. These results suggest that adaptive temporal targets improve behavioral efficiency by progressively redirecting reinforcement pressure toward increasingly efficient successful behavior discovered during learning.

Why efficiency-weighted replay helps. Adaptive temporal targets bias the current PPO update toward efficient successful behavior, but later on-policy updates can still move the policy away from those trajectories. TSIL addresses this by preserving temporally efficient successful trajectories in replay and explicitly increasing their likelihood during optimization. Figure 7 further shows that TSIL redirects PPO updates toward regions with high fast-success-memory likelihood and maintains successful behavior, whereas generic SIL provides weaker alignment and does not reliably preserve success. Landscape construction and additional tasks are provided in Appendices D.3–D.4. Table 1 compares TSIL with TSIL (no efficiency-weighting) ablation, isolating the contribution of efficiency weighting while holding fast-success replay fixed. Removing the efficiency weighting lowers AUC and success rate and increases normalized completion time, showing that prioritizing more efficient successes improves both learning and behavioral efficiency.

Tab. 1: **Efficiency-weighting (EW) ablation in Meta-World.** Without efficiency weighting, TSIL shows lower performance, reported as mean $\pm$ SD across tasks and seeds.

Method	AUC (%) ↑	Success (%) ↑	Norm. time (%) ↓
TSIL (no EW)	57.2 ± 6.6	89.31 ± 5.04	38.6 ± 4.7
TSIL	$\mathbf{63.4 \pm 1.0}$	$\mathbf{95.1 \pm 4.8}$	$\mathbf{33.4 \pm 3.1}$

5 Conclusion & Limitations

We introduced Temporal Self-Imitation Learning (TSIL), which uses temporally efficient successful trajectories as reusable self-supervision through adaptive temporal targets and efficiency-weighted replay. Across 30 long-horizon manipulation and navigation tasks, including disturbed training settings, TSIL improves task success, learning efficiency, behavioral efficiency, and robustness. Our analysis shows that adaptive targets focus learning on efficient success, while replay helps preserve behaviors that unstable on-policy updates might otherwise erase.

Limitations. TSIL does not directly solve initial exploration. It improves exploitation and consolidation after rare successful trajectories appear, so the base learner still needs enough exploration or task guidance to discover success. Temporal efficiency is also only one behavioral signal considered here. If deployment requires diverse modes such as slower, safer, smoother, or more stable behavior, future systems could first learn reliable efficient behavior and then combine temporal targets with additional metrics that reflect those requirements. Finally, TSIL requires storing successful trajectories in memory. We used a small fixed-size replay buffer, whereas more memory-efficient replay and compression could preserve fast successes more effectively.

AI USE STATEMENT

We used an LLM-based tool to polish writing, identify additional relevant literature, and inspect code. The authors reviewed all AI-assisted output and take responsibility for the final manuscript.

REFERENCES

André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. Temporl: Learning when to act. In *International Conference on Machine Learning*, pp. 914–924. PMLR, 2021.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2023.

Jesus Bujalance and Fabien Moutarde. Reward relabelling for combined reinforcement and imitation learning on sparse-reward tasks. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 2565–2567, 2023.

Yu'an Chen, Ruosong Ye, Ziyang Tao, Hongjian Liu, Guangda Chen, Jie Peng, Jun Ma, Yu Zhang, Jianmin Ji, and Yanyong Zhang. Reinforcement learning for robot navigation with adaptive forward simulation time (afst) in a semi-markov model, 2023. URL https://arxiv.org/abs/2108.06161.

Zhixin Chen and Mengxiang Lin. Self-imitation learning for robot tasks with sparse and delayed rewards. In *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 477–482. IEEE, 2021.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tianhong Dai, Hengyan Liu, and Anil Anthony Bharath. Episodic self-imitation learning with hindsight. *Electronics*, 9(10):1742, 2020.

Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1): 219–245, 2000.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Yinsen Jia and Boyuan Chen. Time as a control dimension in robot learning, 2026. URL https://arxiv.org/abs/2511.07654.

Viraj Joshi, Zifan Xu, Bo Liu, Peter Stone, and Amy Zhang. Benchmarking massively parallelized multi-task reinforcement learning for robotics tasks. In *Reinforcement Learning Conference*, 2025. URL https://openreview.net/forum?id=z0MM0y20I2.

Yeo Jin Kim and Min Chi. Time-aware q-networks: Resolving temporal irregularity for deep reinforcement learning. *arXiv preprint arXiv:2105.02580*, 2021.

Yeo Jin Kim and Min Chi. Time-aware deep reinforcement learning with multi-temporal abstraction. *Applied Intelligence*, 53(17):20007–20033, 2023.

John W Krakauer, Alkis M Hadjiosif, Jing Xu, Aaron L Wong, and Adrian M Haith. Motor learning. *Comprehensive Physiology*, 9(2):613–663, 2019. doi: 10.1002/cphy.c170043.

Aravind S. Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. Dynamic action repetition for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Guanzhou Li, Jianping Wu, and Yujing He. Act better by timing: A timing-aware reinforcement learning for autonomous driving. *arXiv preprint arXiv:2406.13223*, 2024.

Yao Li, YuHui Wang, and XiaoYang Tan. Self-imitation guided goal-conditioned reinforcement learning. *Pattern Recognition*, 144:109845, 2023.

Yunfei Li, Yilin Wu, Huazhe Xu, Xiaolong Wang, and Yi Wu. Solving compositional reinforcement learning problems via task reduction. In *International Conference on Learning Representations*, 2021.

Yunfei Li, Tian Gao, Jiaqi Yang, Huazhe Xu, and Yi Wu. Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pp. 12765–12781. PMLR, 2022.

Qian Lin, Bo Tang, Zifan Wu, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Safe offline reinforcement learning with real-time budget constraints. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21127–21152. PMLR, 2023. URL https://proceedings.mlr.press/v202/lin23h.html.

Sha Luo and Lambert Schomaker. Reinforcement learning in robotic motion planning by combined experience-based planning and self-imitation learning. *Robotics and Autonomous Systems*, 170: 104545, 2023.

Sha Luo, Hamidreza Kasaei, and Lambert Schomaker. Self-imitation learning by planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4823–4829. IEEE, 2021.

Yongle Luo, Kun Dong, Lili Zhao, Zhiyong Sun, Chao Zhou, and Bo Song. Balance between efficient and effective learning: Dense2Sparse reward shaping for robot manipulation with environment uncertainty. *arXiv preprint arXiv:2003.02740*, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Anh N. Nhu, Sanghyun Son, and Ming Lin. Time-aware world model for adaptive prediction and control. In *Forty-second International Conference on Machine Learning*, 2025.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*, pp. 4045–4054. PMLR, 2018.

Simon Ramstedt and Chris Pal. Real-time reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 3067–3076, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Seyed Kamyar Seyed Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, and Igor Mordatch. Self-improving embodied foundation models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2509.15155.

Archit Sharma, Ahmed M. Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3292–3308. PMLR, 2023. URL https://proceedings.mlr.press/v229/sharma23b.html.

Sahil Sharma, Aravind Srinivas, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *International Conference on Learning Representations*, 2017.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1–2):181–211, 1999.

Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pp. 6096–6104. PMLR, 2019.

Pierre Vassiliadis, Gerard Derosiere, Cecile Dubuc, Aegryan Lete, Frederic Crevecoeur, Friedhelm C Hummel, and Julie Duque. Reward boosts reinforcement-based motor learning. *iScience*, 24(7):102821, 2021. doi: 10.1016/j.isci.2021.102821.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.

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 *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

A Algorithm and Implementation Details

A.1 Algorithm details

Optimization objective. TSIL uses the same PPO objective as all baselines; temporal target conditioning only changes the observation and reward passed to PPO. For rollout data $\mathcal{T}_i$ collected at training iteration i, define the PPO probability ratio

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid \tilde{o}_t)}{\pi_{\theta_{\text{old}}}(a_t \mid \tilde{o}_t)}. \tag{4}$$

$$\bar{\rho}_t(\theta) = \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon). \tag{5}$$

$$\ell_t^{\text{clip}}(\theta) = -\min(\rho_t(\theta)A_t, \bar{\rho}_t(\theta)A_t)\,. \tag{6}$$

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}_{t\sim\mathcal{T}_i}\left[\ell_t^{\text{clip}}(\theta) + \frac{c_V}{2}\left(V_\psi(\tilde{o}_t) - G_t\right)^2 - c_H\mathcal{H}(\pi_\theta(\cdot \mid \tilde{o}_t))\right]. \tag{7}$$

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{SI}}\mathcal{L}_{\text{SI}}. \tag{8}$$

Here π_θ is the current actor with parameters θ, $\pi_{\theta_{\text{old}}}$ is the rollout policy used to collect $\mathcal{T}_i$, a_t is the action sampled at time step t, and $\tilde{o}_t$ is the temporally augmented observation. The operator $\text{clip}(x, l, u) = \min(\max(x, l), u)$ truncates its first argument to the interval $[l, u]$, ϵ is the PPO clipping threshold, $\bar{\rho}_t$ is the clipped probability ratio, and ℓ_t^{clip} is the per-transition clipped policy loss. A_t and G_t are the on-policy PPO advantage and return estimates, V_ψ is the critic with parameters ψ, c_V and c_H are the value-loss and entropy coefficients, and $\mathcal{H}(\pi_\theta(\cdot \mid \tilde{o}_t))$ is the policy entropy. The expectation $\mathbb{E}_{t\sim\mathcal{T}_i}$ averages over rollout timesteps in $\mathcal{T}_i$. The self-imitation coefficient is λ_{SI}, and the self-imitation loss $\mathcal{L}_{\text{SI}}$ is defined in Eq. (3).

Replay buffer update. The replay buffer is intended to preserve behaviors that show a configuration can be solved efficiently. For each configuration, TSIL therefore keeps at most k successful trajectories with the shortest completion times and discards slower successful trajectories once the replay buffer is full. Given a candidate set $\mathcal{S}$, this update is

$$\text{TopKFastSuccess}(\mathcal{S}, k) = \text{sort}_{T^{\text{succ}}\uparrow}\left(\{\tau \in \mathcal{S} \mid \mathbb{1}_{\text{succ}}(\tau) = 1\}\right)_{1:k}. \tag{9}$$

where $T^{\text{succ}}(\tau)$ is the completion time of trajectory τ. The arrow $T^{\text{succ}}\uparrow$ denotes ascending order, so smaller completion times are ranked first and retained with higher priority.

Fallback replay. For compactness, Algorithm 1 shows the fast-success replay buffer update. In implementation, the replay buffer also maintains a return-ranked fallback pool. When fewer than k successful trajectories are available for the fast-success replay buffer of a configuration, the remaining replay slots are filled by top-return trajectories from the same configuration to stabilize early minibatch updates. These fallback trajectories use the same target-conditioned relabeling as fast successes. Let $|\tau|$ denote the stored length of trajectory τ in replay transitions. Relabeled returns include a value bootstrap for truncated trajectories,

$$\hat{G}_t^\tau = \sum_{j=t}^{|\tau|-1} \gamma^{j-t}\hat{r}_j^\tau + \gamma^{|\tau|-t}(1-d_\tau)V_\psi(\hat{o}_{|\tau|}^\tau), \tag{10}$$

Algorithm 1 Temporal Self-Imitation Learning

Require: configurations $\{\phi_m\}_{m=1}^M$, horizon N, interval Δt, replay buffer size k, PPO epochs E, step size η, SIL coefficients $\lambda_{\text{SI}}, \lambda_{\text{V,SI}}$

$\theta, \psi \leftarrow$ initialize actor and critic; $\mathcal{B}_0(\phi) \leftarrow \emptyset$; $D_1(\phi) \leftarrow T^{\max}$
for iteration $i = 1, 2, \ldots$ **do**
  $\mathcal{T}_i \leftarrow \text{Rollout}(\pi_\theta, D_i)$, with $\tilde{o}_t = (o_t, T_t^{\text{used}}, D_i(\phi))$, $r_t = r_t^{\text{task}} + r_t^{\text{succ}}$ — Eq. (2)
  $\mathcal{B}_i(\phi) \leftarrow \text{TopKFastSuccess}(\mathcal{B}_{i-1}(\phi) \cup \mathcal{T}_i(\phi), k)$ — Eq. (9)
  $\hat{\mathcal{B}}_i(\phi) \leftarrow \text{Relabel}(\mathcal{B}_i(\phi), D_i)$; compute $\hat{G}_t^\tau$, $A_t^+(\tau)$, $w_i(\tau)$ — Sec. 3.3, Eq. (10)
  for $e = 1, \ldots, E$ **do**
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{PPO}}(\theta, \psi; \mathcal{T}_i) + \lambda_{\text{SI}}\mathcal{L}_{\text{SI}}(\theta, \psi; \hat{\mathcal{B}}_i)$ — Eq. (8)
    $(\theta, \psi) \leftarrow (\theta, \psi) - \eta\nabla_{\theta,\psi}\mathcal{L}$
  end for
  $D_{i+1} \leftarrow \text{TargetUpdate}(D_i, \mathcal{T}_i)$ — Eq. (1)
end for

Here $d_\tau = 1$ for terminal success or failure, and $d_\tau = 0$ for truncation without success; only the latter receives a value bootstrap.

On-policy training clarification. The PPO loss is computed only from the freshly collected on-policy rollout batch $\mathcal{T}_i$. Stored trajectories do not enter the PPO likelihood ratio or an off-policy Bellman backup. They are used only through the auxiliary self-imitation loss in Eq. (3). The value term in $\mathcal{L}_{\mathrm{SI}}$ regresses the critic toward relabeled Monte Carlo or bootstrapped replay-buffer returns when the positive return-value gap is nonzero. This is auxiliary value supervision rather than a one-step off-policy Bellman target. Thus, TSIL remains an on-policy PPO method augmented with a small fast-success memory for self-imitation.

A.2 Implementation details

Parallel-environment implementation. Temporal efficiency is configuration-conditioned: a short completion time for a simple configuration may not be enough for a hard configuration. We therefore fix the configuration within each parallel environment: each environment keeps a persistent configuration ϕ across resets, while configurations are randomly sampled independently across environments. TSIL maintains the adaptive temporal target $D(\phi)$ and replay buffer independently for each parallel environment.

Unusually fast successes. Based on our parallel-environments setting, an unusually fast success tightens only the temporal target of its own configuration, leaving the targets of other configurations unchanged. Its trajectory is retained among the fastest successes and replayed to encourage the policy to reproduce the behavior. Moreover, since the target remains soft, even slower successes still receive the base success reward plus the temporal bonus in Eq. (2). For successful completion times above the target, this bonus increases as completion time approaches the target, so matching the exact fastest observed time is not required to receive a learning signal. These mechanisms preserve supervision even when the fastest success is difficult to repeat.

Replay relabeling. During replay, stored trajectories keep their original observations, actions, and elapsed-time channel, but the target channel is relabeled using the current temporal target $D(\phi)$. The task reward is unchanged, and the terminal temporal success reward is recomputed under the current target. Replay returns use the relabeled rewards; truncated fallback trajectories additionally use the value bootstrap in Eq. (10).

Replay sampling. For each configuration, TSIL selects up to k stored trajectories, prioritizing the fastest successes and filling unused slots with the highest-return trajectories. TSIL screens all valid transitions using the positive return-value gap and samples up to n positive transitions per trajectory without replacement, with probability proportional to the gap. The replay set is constructed once per PPO update and reused across E epochs. With N_{env} parallel environments, it contains at most knN_{env} transitions and can be smaller when fewer positive transitions are available at early training.

Evaluation temporal target inference. During the evaluation stage for unseen configurations, TSIL uses the mean temporal target of up to five nearest successful training configurations from the same task. If none of the task's training configurations has succeeded, TSIL uses the maximum episode horizon. The inferred target remains fixed throughout evaluation. Configuration representations are provided by the task environment described below.

MiniGrid configuration. The configuration ϕ includes the initial symbolic grid, agent position and orientation, and mission where applicable. Within each task, distance combines the fraction of grid cells with different object/color/state encodings, the agent's Manhattan displacement normalized by grid dimensions, and penalties for orientation and mission mismatches.

Meta-World configuration. The configuration ϕ is represented by the task's object/goal randomization vector. Within each task, distance is Euclidean distance between these vectors after scaling each feature by its range across training configurations.

Normalization. All methods use the same normalization pipeline. Physical observation and state features use running mean–standard-deviation normalization. Temporal features, including elapsed time and the current temporal target, are excluded from running-statistic normalization and are instead scaled by $T^{\max}$. Rewards are rescaled using per-task statistics of a backward-discounted reward trace, $z_t = \gamma z_{t-1} + r_t$, maintained separately for each environment and reset after each

episode. Each reward is divided by the trace's running standard deviation. This rescales individual rewards without subtracting a mean; it does not directly normalize forward returns. Relabeled replay rewards use the current on-policy scale without updating its statistics. When value normalization is enabled, PPO returns, critic predictions, and relabeled replay-buffer returns are mapped into the same normalized value space before value losses or the positive return-value gap are computed. PPO advantages are normalized per task before the policy gradient update. For replay actor updates, the positive weights $w_i(\tau)A_t^+(\tau)$ are divided by the standard deviation of their positive entries (without mean subtraction) and capped at 1; this normalization does not rescale the replay critic loss.

Optimization hyperparameters. The PPO update, rollout length, minibatch size, discounting, GAE parameter, value coefficient, entropy coefficient, and learning rate schedule are shared with all baselines. The method-specific settings are the replay buffer size, replay sampling budget, and auxiliary loss coefficients λ_{SI} and $\lambda_{\mathrm{V,SI}}$. The complete shared PPO settings and replay-specific settings are listed in Tables 4 and 5.

B Training and Evaluation Setup

B.1 Benchmark and agent setup

Tasks. We evaluate each method on 15 MiniGrid tasks and 15 Meta-World robot manipulation tasks, with an independently trained policy for each task and three random seeds per task. MiniGrid contains Empty, GoToObject, DoorKey, SimpleCrossing, LavaGap, MultiRoom, DynamicObstacles, FourRooms, Unlock, UnlockPickup, BlockedUnlockPickup, KeyCorridor, RedBlueDoors, Fetch, and GoToDoor. The Meta-World task ids are $\{0, 1, 2, 3, 14, 18, 21, 22, 27, 28, 29, 30, 43, 44, 46\}$. Figures 8 and 9 show the MiniGrid and Meta-World task suites, respectively.

MiniGrid agent design. The MiniGrid policy receives the local 7×7 symbolic image and compact mission-target features. Separate actor and critic encoders use three stride-one, unpadded 2×2 convolutions with 16, 32, and 64 channels. The first two convolutions use ReLU and the third uses tanh, producing $64 \times 4 \times 4 = 1024$ image features. These features are concatenated with the auxiliary inputs and passed to five-hidden-layer MLPs with widths $[512, 256, 128, 64, 32]$, tanh activations, and LayerNorm. The actor outputs categorical action logits and the critic outputs a scalar value. Temporal-target methods additionally receive elapsed steps and the active temporal target, while these inputs are masked for non-temporal baselines.

Meta-World agent design. The policy observes robot proprioceptive features and task state, including end-effector position, gripper openness, object poses, target position, and a one-step memory of robot and object state. For temporal-target methods, this observation is augmented with elapsed time and the adaptive temporal target as described in Sec. 3.1. The action is a four-dimensional continuous command consisting of a Cartesian end-effector position delta and a gripper command, which is executed by the Franka controller in Isaac Gym. The actor and critic are five-hidden-layer MLPs with hidden sizes $[512, 256, 128, 64, 32]$, tanh activations, and LayerNorm. The actor predicts a mean $\mu_\theta(\tilde{o}_t)$ and uses a state-independent diagonal covariance,

$$a_t \sim \pi_\theta(\cdot \mid \tilde{o}_t), \qquad \pi_\theta(\cdot \mid \tilde{o}_t) = \mathcal{N}\big(\mu_\theta(\tilde{o}_t), \mathrm{diag}(\sigma^2)\big), \tag{11}$$

where $\sigma \in \mathbb{R}^4$ is the action-standard-deviation vector parameterized by a learned state-independent log-standard-deviation vector. The operator $\mathrm{diag}(\sigma^2)$ forms a diagonal covariance matrix with entries σ_j^2 on the diagonal. The policy observation can be obtained on a real setup from robot state estimation and calibrated perception or tracking. Elapsed time and the temporal target are directly available scalar signals.

Meta-World controller design. The high-level policy outputs Cartesian end-effector position deltas and a gripper command at 60 Hz. The arm command is converted by damped differential inverse kinematics into target joint positions. These joint targets can be tracked by a low-level joint impedance controller at 250 Hz, which converts joint-position errors into torque commands. The differential IK module limits arm-joint commands using nominal velocity bounds of 2.175 rad/s for joints 1–4 and 2.610 rad/s for joints 5–7, and includes an exponential moving average smoothing option for joint increments, which can be used for future real-robot deployment to attenuate abrupt command changes and reduce oscillatory motion. This hierarchy keeps the learned interface close

to standard real-robot control stacks by delegating robot-specific dynamics to IK and impedance control.

B.2 Disturbed training settings

Disturbance experiments compare SIL, ATTL, and TSIL while perturbing one training factor and leaving the remaining preprocessing and loss-scaling settings unchanged. For MiniGrid, we evaluate policy-gradient noise and increased PPO clip ratios. For Meta-World, we additionally evaluate dense-reward dropout. Table 2 summarizes the applicable disturbance classes.

Policy gradient noise. We add Gaussian noise to the PPO actor gradient and restore its original global norm before combining it with the other gradients:

$$\begin{aligned} \epsilon_{\alpha,b} &= \alpha_{\text{noise}}\,\mathrm{RMS}(g_{\text{PPO},b})z_b, \qquad z_b \sim \mathcal{N}(0, I), \\ g_{\text{actor}} &= \frac{\|g_{\text{PPO}}\|_2}{\|g_{\text{PPO}} + \epsilon_\alpha\|_2}(g_{\text{PPO}} + \epsilon_\alpha) + g_{\text{replay}} + g_{\text{other}}. \end{aligned} \tag{12}$$

Here b indexes a parameter tensor, $\alpha_{\text{noise}} \in \{5, 10, 20\}$ is the dimensionless noise scale, and $\mathrm{RMS}(g) = \sqrt{\sum_{j=1}^{d} g_j^2/d}$ for a tensor with d entries. Noise entries are sampled independently; the norms span all PPO actor parameters. The PPO gradient includes entropy regularization, g_{replay} includes the replay coefficient, and g_{other} contains other actor regularization gradients. Replay and other gradients receive no injected noise. If norm restoration is undefined, we retain the clean PPO gradient. The combined gradient undergoes the usual clipping and optimizer update.

Dense reward dropout. We dropped dense shaping rewards episode-wise with dropout probability p, while keeping the target-conditioned success reward. The dropout mask was resampled when an environment finished an episode. This tested whether replay could help policy training when shaped guidance became intermittent or unreliable.

Large PPO clip ratio. We increased the PPO clip ratio value beyond the default setting, allowing larger policy updates per minibatch and weakening PPO's trust-region-style constraint. This sweep tested whether self-imitation could stabilize these more aggressive updates while preserving training efficiency.

Tab. 2: **Disturbed training settings.** Each experiment varies one disturbance type while keeping the remaining training configuration fixed.

Disturbance type	MiniGrid values	Meta-World values
Policy gradient noise scaling	$5, 10, 20$	$5, 10, 20$
Dense reward dropout probability	—	$0.4, 0.6, 0.8$
PPO clip ratio	$0.3, 0.5, 0.7$	$0.3, 0.5, 0.7$

B.3 Baselines and hyperparameters

Within each benchmark, all methods use PPO as the base optimizer and share the same agent architecture, rollout collection, preprocessing, and other training components. We used the baseline set described in Sec. 4. All methods use the same base success-reward coefficient within each benchmark: $r_{\text{succ}} = 100$. Tables 3 and 4 list the benchmark-specific training configurations.

Self-imitation methods use the same auxiliary coefficients and perform joint updates every PPO minibatch. Table 5 lists these replay-specific settings.

B.4 Primary metric computation

Training success rate is the maximum rolling success rate recorded during the final 10% of the training budget. Training AUC is the trapezoidal integral of the rolling success curve over the full interaction budget, divided by that budget. Successful episode count is the total number of successful episodes collected during training.

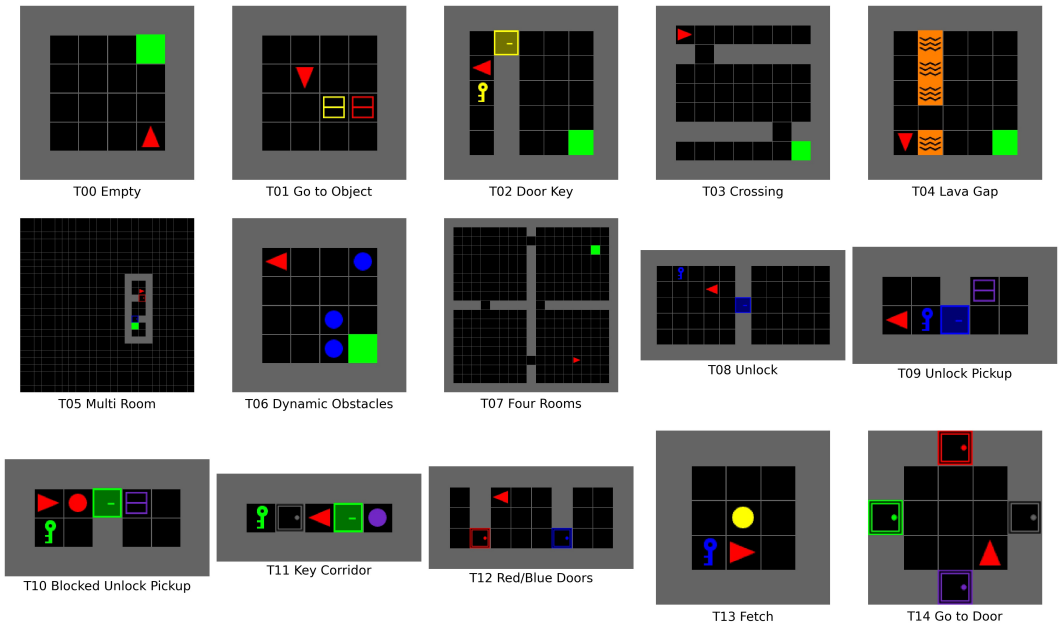

Fig. 8: **MiniGrid task suite.** Representative layouts for the 15 MiniGrid tasks used in our evaluation, covering navigation, obstacle avoidance, door-and-key interaction, object pickup, and goal-conditioned object interaction.

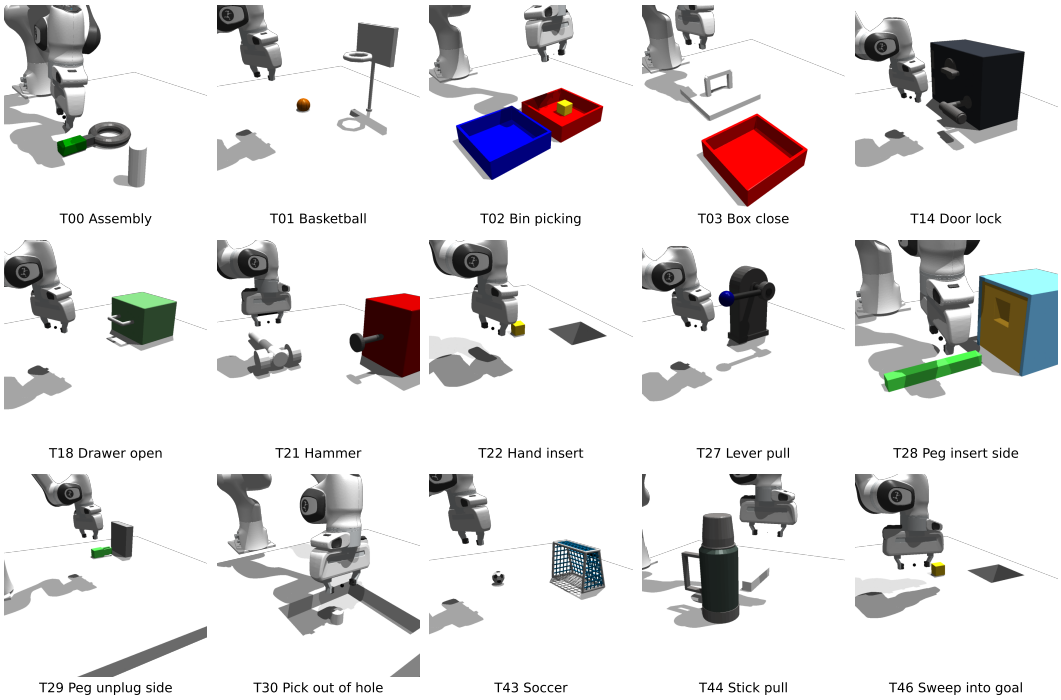

Fig. 9: **Meta-World task suite.** Representative configurations for all 15 Meta-World manipulation tasks.

Tab. 3: **MiniGrid training configuration.** Episode horizons are fixed separately for each task.

Item	Value
Parallel environments	32
Episode horizon	100–720 steps, task dependent
Total training budget	640,000 steps
PPO rollout length	32 steps
Discount / GAE parameter	$\gamma = 0.995$ / $\lambda = 0.95$
Learning rate	5×10^{-4}
Success reward scale	$r_{\mathrm{succ}} = 100$
PPO clip ratio	0.2
PPO epochs / minibatch size	5 / 128
Value coefficient	4
Entropy coefficient	0.005
Evaluation trials per task per seed	2000
Evaluation configurations per task per seed	32
Evaluation checkpoint	best final-10% training success rate

Tab. 4: **Meta-World training configuration.** Values apply unless explicitly swept in the disturbance experiments.

Item	Value
Parallel environments	6400
Episode horizon	150 steps
Total training budget	50×10^6 steps
PPO rollout length	32 steps
Discount / GAE parameter	$\gamma = 0.995$ / $\lambda = 0.95$
Learning rate	5×10^{-4}
Success reward scale	$r_{\mathrm{succ}} = 100$
PPO clip ratio	0.2
PPO epochs / minibatch size	5 / 20480
Value coefficient	4
Entropy coefficient	0.005
Evaluation trials per task per seed	2000
Evaluation configurations per task per seed	2000
Evaluation checkpoint	best final-10% training success rate

For an episode e with duration T_e, success indicator s_e, and fixed task horizon $T_e^{\max}$, normalized completion time is $s_e T_e / T_e^{\max} + (1 - s_e)$. During training, we average this quantity over the same rolling window used for success rate and report its value at the checkpoint with the maximum rolling success rate within the final 10% of training. Evaluation success rate is computed from the reported policy using the benchmark evaluation protocol and the checkpoint with the maximum final-10% training success rate.

Rolling statistics and evaluation actions. The rolling success rate averages the most recent completed episodes of each task, using a buffer of $\max(5N_{\mathrm{env}}, 50)$ episodes. With independently trained

Tab. 5: **SIL-specific replay configuration.** SIL stores high-return trajectories, while TSIL prioritizes fast successful trajectories and fills unused replay slots with high-return trajectories. Both methods use the same gap-proportional sampler.

Item	SIL	TSIL
Replay source	high-return	fast-success first with high-return as fallback
Sample selection	positive-gap proportional, without replacement	positive-gap proportional, without replacement
Trajectory weighting	none	efficiency weighting
Per-env. replay buffer top-k	5	5
Maximum sampled transitions per selected trajectory, n	32	32
Replay samples per PPO update	$\leq knN_{\mathrm{env}}$	$\leq knN_{\mathrm{env}}$
Policy replay coefficient	$\lambda_{\mathrm{SI}} = 0.05$	$\lambda_{\mathrm{SI}} = 0.05$
Value replay coefficient	$\lambda_{\mathrm{V,SI}} = 0.05$	$\lambda_{\mathrm{V,SI}} = 0.05$

policies, this gives 160 episodes for MiniGrid and 32,000 for Meta-World. Evaluation samples actions from the learned categorical policy in MiniGrid and Gaussian policy in Meta-World.

C STANDARD-TRAINING RESULTS

Figures 10–13 expand main-text Fig. 4 with standard-training learning curves for every task. Lines show means across seeds, and shaded regions show ± one standard deviation (SD) across seeds.

C.1 MINIGRID

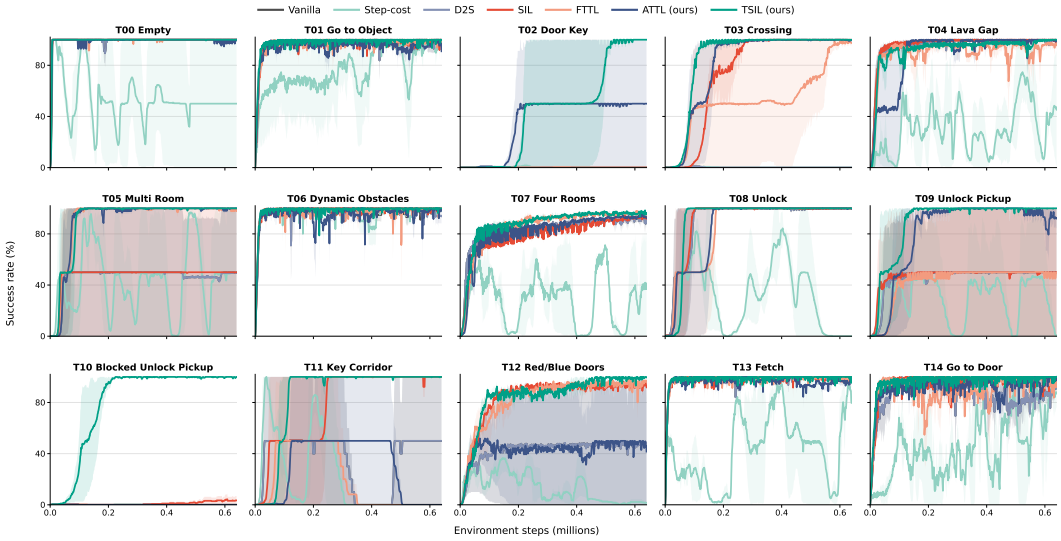

Fig. 10: **Per-task MiniGrid success rate.** Success rate over training for all 15 MiniGrid tasks. Higher values indicate more reliable task completion.

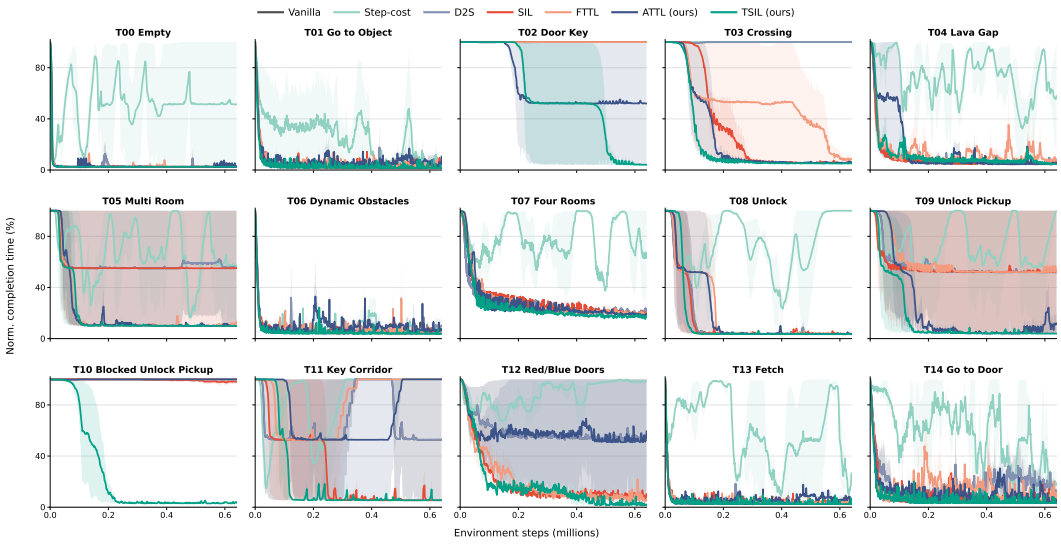

Fig. 11: **Per-task MiniGrid normalized completion time.** Normalized completion time over training for all 15 MiniGrid tasks. Failures receive the full task horizon, and lower values indicate more reliable and temporally efficient completion.

C.2 META-WORLD

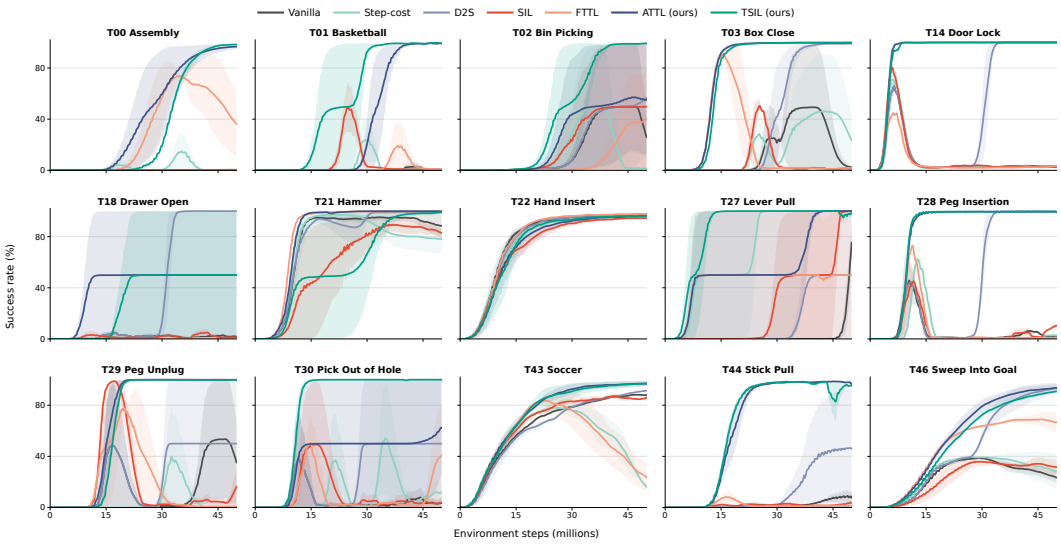

Fig. 12: **Per-task Meta-World success rate.** Success rate over training for all 15 Meta-World tasks. Higher values indicate that the policy completes the task more reliably.

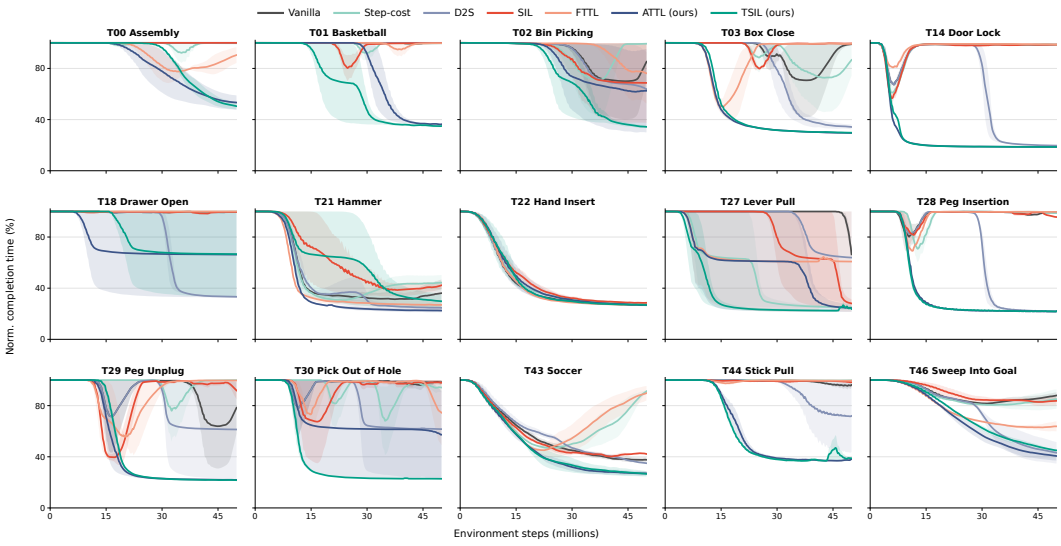

Fig. 13: **Per-task Meta-World normalized completion time.** Normalized completion time over training for all 15 Meta-World tasks. Failures receive the full task horizon, and lower values indicate more reliable and temporally efficient completion.

D Additional Analysis

D.1 Learning-signal metric computation

We provide detailed definitions for the diagnostic metrics used in Sec. 4.3. These metrics quantify where positive-advantage mass concentrates and how self-imitation affects revisitation of stored behavior.

Learning-signal diagnostics. For the learning-signal analysis, completed episodes were grouped into bins by normalized completion time and discounted task-reward return. As shown in Fig. 6, cell color shows empirical success rate in each bin, while the red overlay gradient shows the fraction of total positive-advantage mass assigned to that bin. Let $\mathcal{E}$ be the set of completed training episodes, and let $b \subset \mathcal{E}$ denote the subset of episodes assigned to a given visualization bin. We define

$$\mathrm{Mass}(b) = \frac{\sum_{\tau \in b} \sum_{t \in \tau} [\hat{A}_t^{\mathrm{PPO}}(\tau)]_+}{\sum_{\tau \in \mathcal{E}} \sum_{t \in \tau} [\hat{A}_t^{\mathrm{PPO}}(\tau)]_+}, \tag{13}$$

where $[x]_+ = \max(x, 0)$ and $\hat{A}_t^{\mathrm{PPO}}(\tau)$ is the advantage after the same normalization used for the PPO policy update.

For the summary bars on the right, let $m_\tau = \sum_{t \in \tau} [\hat{A}_t^{\mathrm{PPO}}(\tau)]_+$ be the positive-advantage mass of episode τ, and let $M = \sum_{\tau \in \mathcal{E}} m_\tau$ be the total positive-advantage mass over the completed episodes. We also let $s_\tau \in \{0, 1\}$ be the success indicator, $\bar{T}_\tau = T_\tau / T^{\max}$ be normalized completion time, and $G_\tau^{\mathrm{task}} = \sum_{t \in \tau} \gamma^t r_t^{\mathrm{task}}$ be the discounted return from the task-reward component. Let $G_{\min}^{\mathrm{task}} = \min_{\tau \in \mathcal{E}} G_\tau^{\mathrm{task}}$, $G_{\max}^{\mathrm{task}} = \max_{\tau \in \mathcal{E}} G_\tau^{\mathrm{task}}$, and define the normalized task return as

$$\rho_\tau = \frac{G_\tau^{\mathrm{task}} - G_{\min}^{\mathrm{task}}}{\max(G_{\max}^{\mathrm{task}} - G_{\min}^{\mathrm{task}}, \epsilon)}, \tag{14}$$

where ϵ is a small numerical constant. The four aggregate mass fractions are defined as:

$$F_{\mathrm{success}} = \frac{1}{M} \sum_{\tau \in \mathcal{E}} m_\tau s_\tau, \tag{15}$$

$$F_{\mathrm{fastsuccess}} = \frac{1}{M} \sum_{\tau \in \mathcal{E}} m_\tau s_\tau (1 - \bar{T}_\tau), \tag{16}$$

$$F_{\mathrm{slowfailure}} = \frac{1}{M} \sum_{\tau \in \mathcal{E}} m_\tau (1 - s_\tau) \bar{T}_\tau, \tag{17}$$

$$F_{\mathrm{rewarddistracted}} = \frac{1}{M} \sum_{\tau \in \mathcal{E}} m_\tau \rho_\tau \bar{T}_\tau, \tag{18}$$

If $M = 0$, all fractions are set to zero. F_{success} measures how much positive-advantage mass comes from successful episodes, indicating whether the update reinforces task completion. $F_{\mathrm{fastsuccess}}$ further weights successful episodes by temporal efficiency, indicating whether the update concentrates on earlier successes. $F_{\mathrm{slowfailure}}$ measures positive-advantage mass assigned to late unsuccessful trajectories, which indicates whether PPO still reinforces horizon-consuming failures. $F_{\mathrm{rewarddistracted}}$ measures positive-advantage mass assigned to late trajectories with high dense task return, indicating whether shaped rewards attract updates toward slow reward-seeking behavior rather than compact completion.

D.2 Additional learning-signal diagnostics

Figures 14–18 extend the analysis in main-text Fig. 6 to five additional Meta-World tasks. Each row is a method; the first four columns divide training into four equal interaction intervals. Cell color shows empirical success rate over normalized completion time and normalized task return, while the red overlay shows the fraction of positive-advantage mass. The episode-count map and summary bars show how each method distributes its update signal across successful, fast-successful, reward-distracted, and slow-failure behavior; metric definitions are in Sec. D.1.

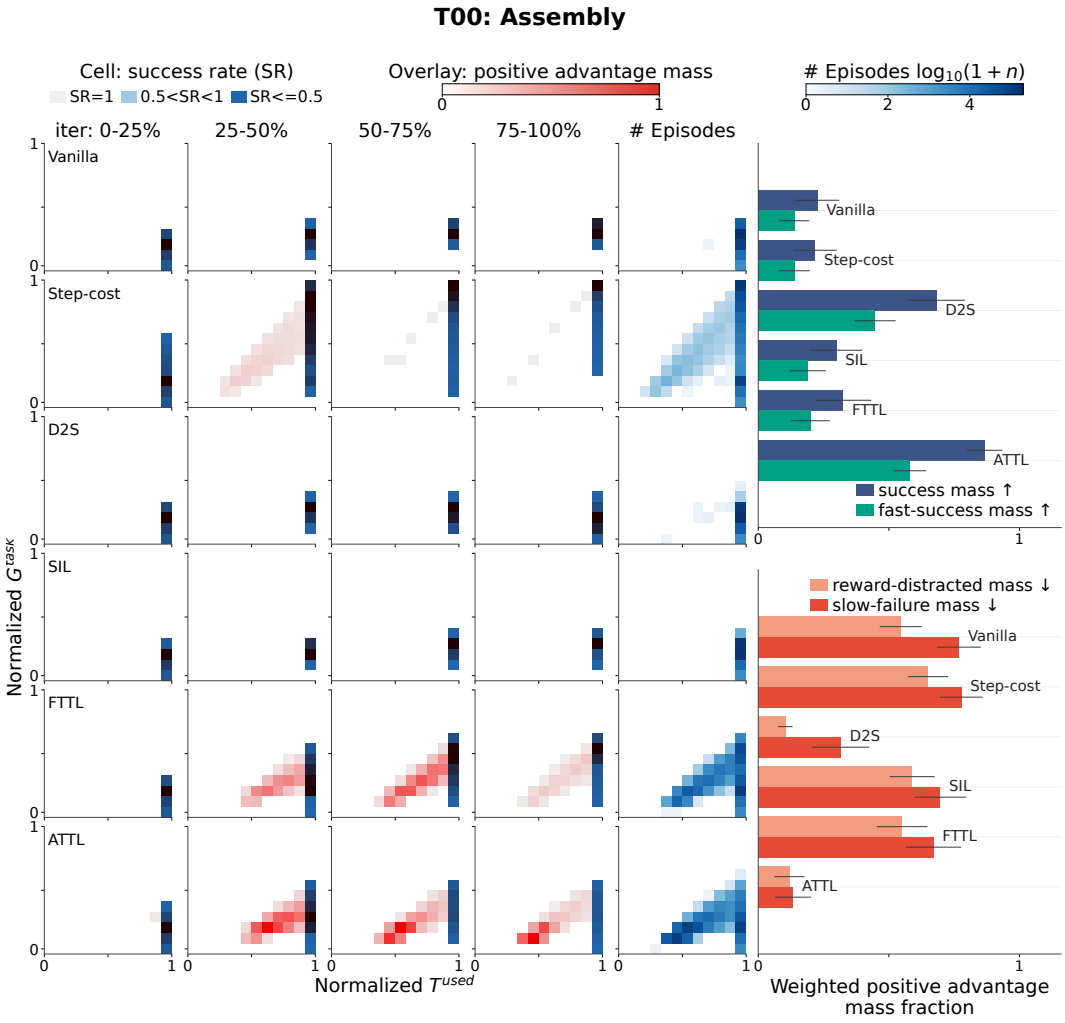

Fig. 14: **Learning-signal analysis for T00: Assembly.** The panels use the same construction and metrics as Fig. 6.

Learning-signal diagnostics (continued).

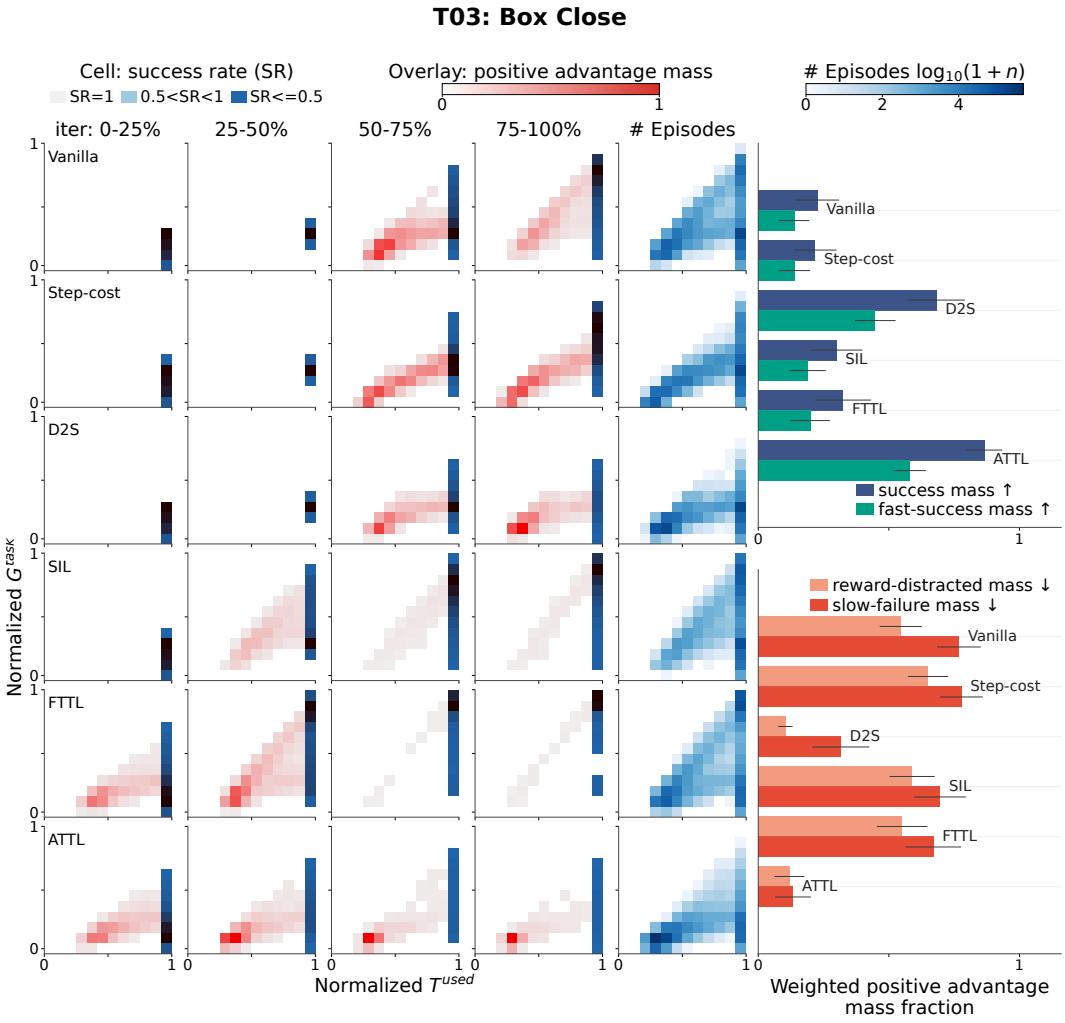

Fig. 15: **Learning-signal analysis for T03: Box Close.** The panels use the same construction and metrics as Fig. 14.

Learning-signal diagnostics (continued).

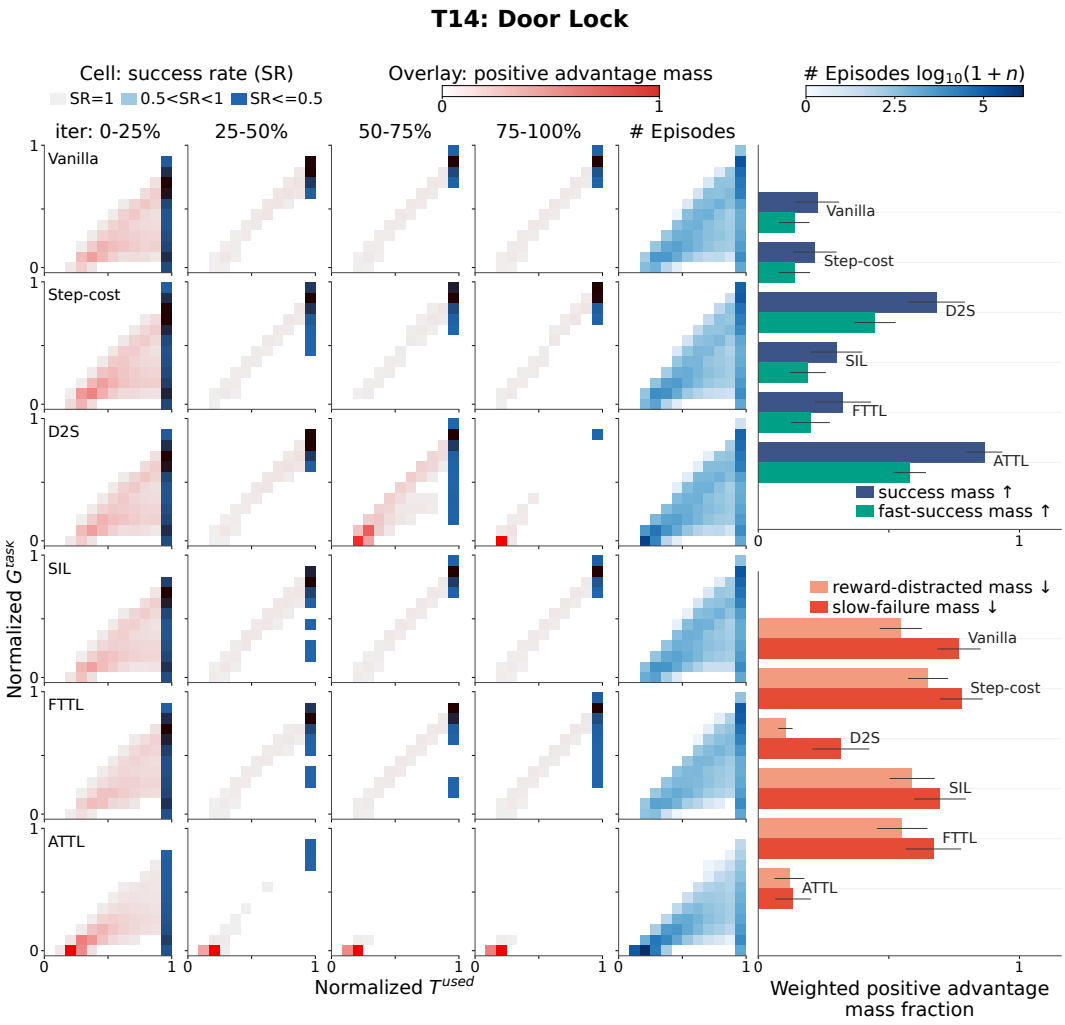

Fig. 16: **Learning-signal analysis for T14: Door Lock.** The panels use the same construction and metrics as Fig. 14.

Learning-signal diagnostics (continued).

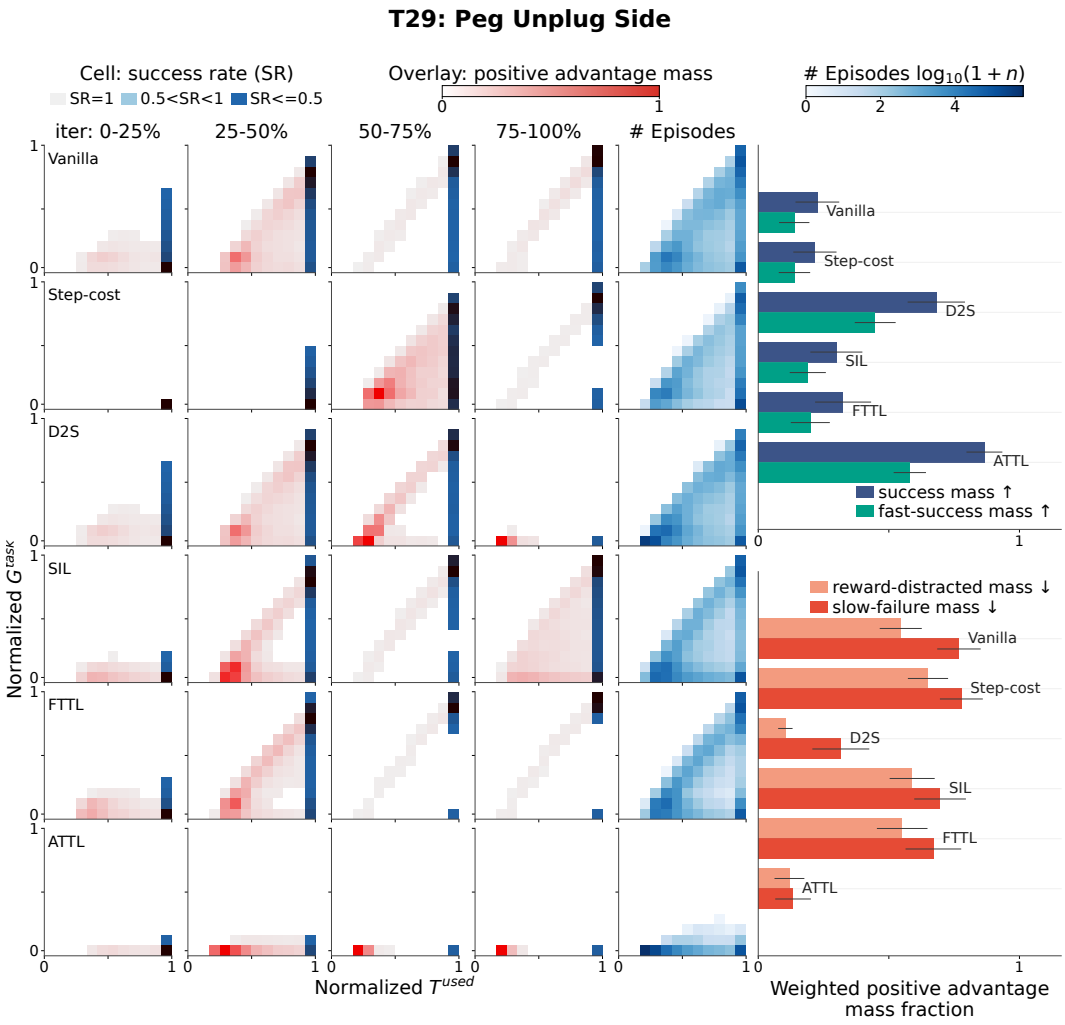

Fig. 17: **Learning-signal analysis for T29: Peg Unplug Side.** The panels use the same construction and metrics as Fig. 14.

Learning-signal diagnostics (continued).

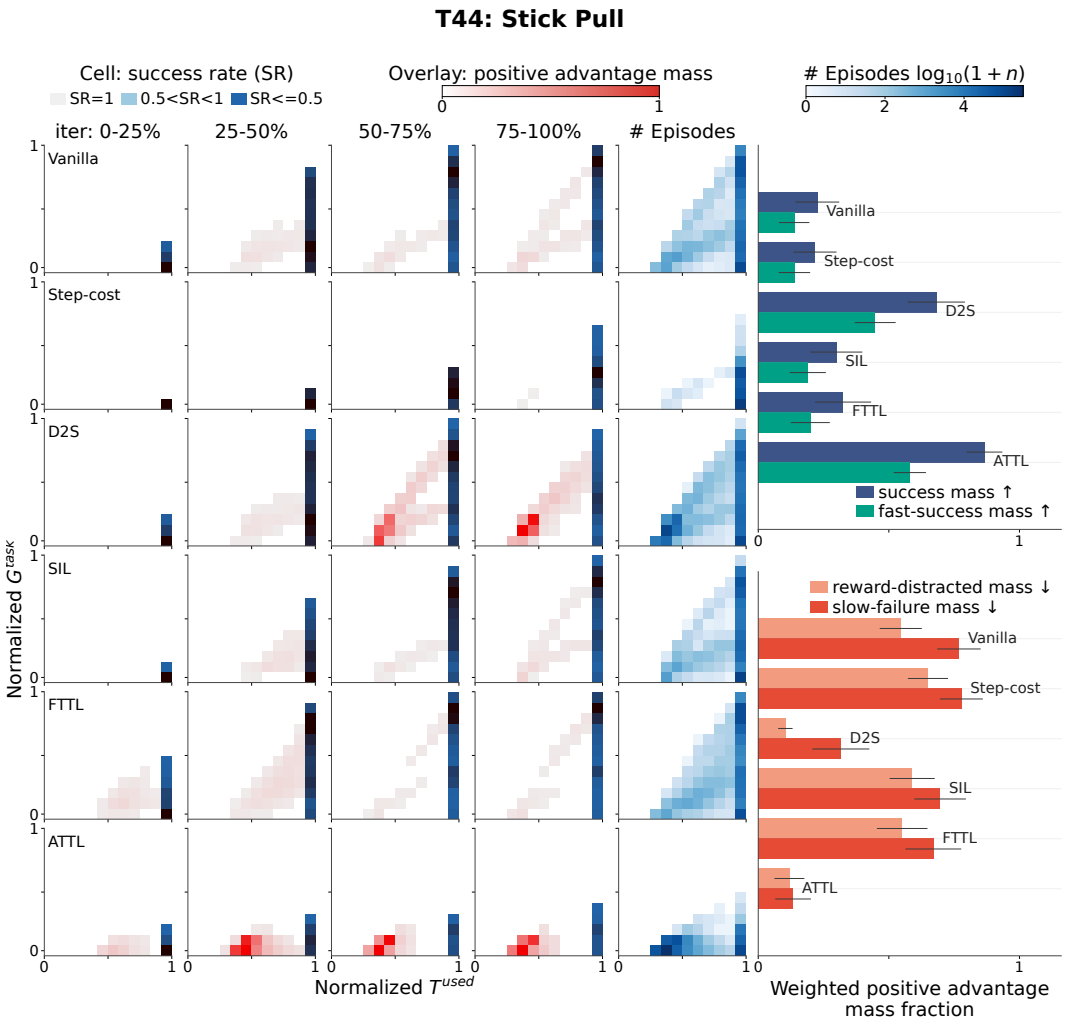

Fig. 18: **Learning-signal analysis for T44: Stick Pull.** The panels use the same construction and metrics as Fig. 14.

D.3 POLICY-UPDATE LANDSCAPE CONSTRUCTION

Main-text Fig. 7 plots mean action log probability on fixed fast-success memory, averaged first over sampled transitions per trajectory, then over trajectories. The horizontal axis follows the negative PPO policy-loss gradient; the vertical axis follows the orthogonal component of the memory log-probability gradient, both normalized. SIL records this memory only for diagnostics, not replay. Surfaces evaluate parameter perturbations in this plane; normalized PPO and replay-combined directions are projected as arrows indicating direction, not step size. Each surface is independently rescaled to $[0, 1]$, so colors indicate within-panel relative likelihood. Black points mark current policies; percentages report rollout success. Panels show 25%, 50%, 75%, and 100% of training.

D.4 ADDITIONAL POLICY-UPDATE LANDSCAPES

These plots extend main-text Fig. 7. Each task occupies one row: the first four panels show PPO and TSIL updates, and the last four show SIL updates. These landscapes suggest that TSIL better preserves fast-success behavior. This improves learning efficiency by making better use of scarce early fast-successes and stabilizes training by reducing the forgetting of efficient successful behaviors.

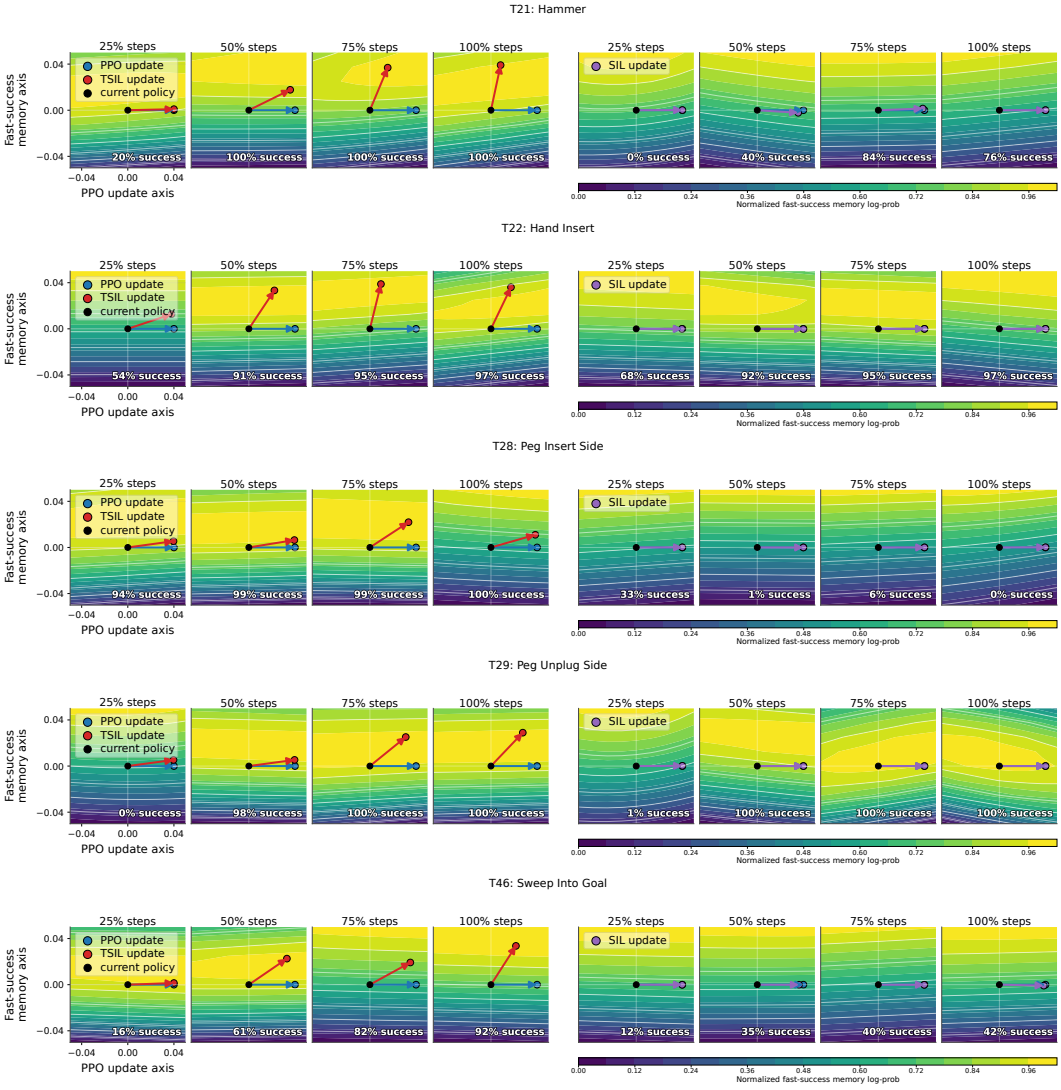

Fig. 19: **Additional policy-update landscapes.** Each group of four panels shows 25%, 50%, 75%, and 100% of training. The landscapes suggest stronger preservation of fast-success behavior with TSIL than with SIL.