WORLD–ACTION COUPLING AND COLLECTION-MATCHED OPTIMIZATION FOR ONLINE MULTI-TASK REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Purely online multi-task reinforcement learning (MTRL) couples a global optimization clock to task-local data streams. With all-task updates at a fixed global frequency, task-wise update-to-data ratios can increase with the number of tasks. We introduce **WAM-MTRL-VE**, which couples latent future prediction to policy learning while matching task optimization to online data acquisition. The actor and an ensemble world model share a task-conditioned representation, and short imagined rollouts provide multi-step reward gradients to the actor through a frozen-model computation path. At each optimization tick, collection-matched stochastic optimization samples one replay-ready task according to the collection weights. Once all replay buffers are ready, matching update and collection probabilities recovers the nominal single-task update-to-data ratio in expectation while interleaving tasks. The entire agent is trained from scratch using online experience, without demonstrations or offline pretraining. On Meta-World, WAM-MTRL-VE reaches $96.2\% \pm 1.6\%$ success on MT10 after 1M total interactions per run, averaged over five training seeds. On MT50, a checkpoint selected from one 12M-interaction training run achieves $82.1\% \pm 0.5\%$ success across five independently seeded evaluation suites.

1 INTRODUCTION

Multi-task reinforcement learning (MTRL) seeks to solve multiple tasks with a single policy by sharing representations and experience (D'Eramo et al., 2020; Sodhani et al., 2021). World models provide a complementary source of policy guidance by predicting the consequences of actions (Hafner et al., 2025; Hansen et al., 2024). In purely online MTRL, however, the policy, world model, and task-wise replay distributions must all develop from ongoing interaction. Effective model guidance therefore requires two forms of alignment: predicted futures must influence action learning, and the optimization rate of each task must track the rate at which that task provides new data. We study both requirements within a single task-conditioned policy.

The second requirement is violated by a conventional all-task update schedule. Suppose that every task is optimized once after every k global environment interactions, while task τ contributes a fraction p_τ of the collected transitions. Its task-wise update-to-data ratio is approximately $\mathrm{UTD}_\tau \approx 1/(kp_\tau)$. Under uniform collection over N tasks, this becomes N/k, compared with $1/k$ in the corresponding single-task setting. Thus, keeping the global update frequency fixed causes each task to reuse its replay data increasingly often as the task set grows. When learned uncertainty and novelty support exploration, repeated fitting on slowly replenished replay may attenuate these signals before reliable successful behavior emerges. We use *discovery deadlock* to describe this observable pattern. It is related to primacy bias in reinforcement learning (Nikishin et al., 2022; Qiao et al., 2024), but focuses on a task-level collection–optimization mismatch. Because scheduling also changes actor, critic, and representation learning, we treat exploration-signal attenuation as a candidate pathway rather than an assumed mediator.

We introduce **WAM-MTRL-VE**, a purely online framework that makes predicted futures actionable while matching task optimization to online data acquisition. First, the policy and an ensemble latent world model read a shared task-conditioned representation. Short imagined rollouts augment the

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.

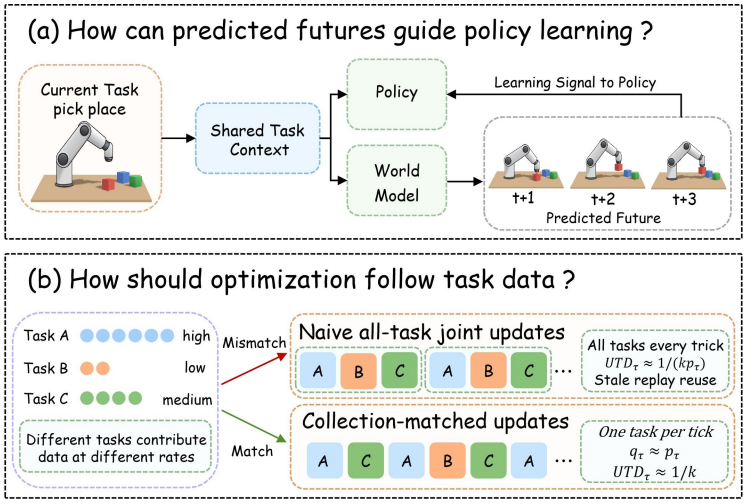

Figure 1: **Motivation. (a)** Predicted futures directly guide the actor. **(b)** Unlike all-task joint updates, our stochastic update matches task-update probabilities q_τ to collection probabilities p_τ, avoiding task-wise UTD inflation.

real-data SAC actor objective with multi-step predicted-reward gradients, while model-side parameters remain fixed along this actor gradient path. Second, collection-matched stochastic optimization samples one replay-ready task at each update tick according to the same progress-dependent weights used for environment collection. If task τ is collected with probability p_τ and updated with probability q_τ, then $\mathrm{UTD}_\tau \approx q_\tau/(kp_\tau)$. Matching q_τ to p_τ therefore recovers the nominal single-task ratio $1/k$ after replay warm-up while interleaving task updates. Value expansion makes model predictions actionable, whereas collection matching prevents task-wise optimization from outrunning online data acquisition. Ensemble disagreement and random network distillation support exploration, and successful online trajectories are reused for actor–critic consolidation.

Our framework builds on complementary lines of prior work. Differentiating policy objectives through learned dynamics is established in stochastic value gradients, model-augmented actor–critic methods, and latent imagination agents (Heess et al., 2015; Clavera et al., 2020; Hafner et al., 2020). PWM applies model gradients to policies using an offline-pretrained multi-task world model (Georgiev et al., 2025). Within MTRL, PaCo and MOORE structure parameter or representation sharing (Sun et al., 2022; Hendawy et al., 2024), while HyMTRL changes optimization after tasks become mastered (He et al., 2026). MoW instead scales predictive capacity through modular latent dynamics (Zhang et al., 2026). Our focus is jointly learning the model and policy from online interaction while explicitly matching task-wise updates to the data distribution that supports them.

We evaluate WAM-MTRL-VE on Meta-World MT10 and MT50 (Yu et al., 2020b). On MT10, it reaches $96.2\% \pm 1.6\%$ success after 1M total interactions across five training seeds. On MT50, a checkpoint from one 12M-interaction training run achieves $82.1\% \pm 0.5\%$ success across five independently seeded evaluation suites. Controlled MT10 comparisons evaluate joint, freshness-gated, uniform single-task, and collection-matched update schedules. A same-seed schedule–intrinsic factorial shows that our schedule outperforms naive all-task updates at the 1M endpoint both with and without intrinsic rewards. Comparisons with uniform single-task updates further evaluate distribution matching at the same task-minibatch budget. Task-level diagnostics characterize the co-occurrence of exploration-signal attenuation and limited hard-task success. Additional ablations evaluate value-expansion routing and success replay.

Our contributions are threefold. First, we characterize schedule-induced task-wise UTD inflation in purely online MTRL and introduce collection-matched stochastic optimization to align task updates with online data acquisition. Second, we integrate actor-side differentiable value expansion with a shared task-conditioned latent model, allowing imagined rewards to shape the policy without updating the model through the actor loss. Third, we evaluate the framework on Meta-World MT10 and MT50 with controlled MT10 studies of update scheduling, actor-gradient routing, and success replay, including a same-seed factorial analysis of scheduling and intrinsic rewards.

2 Related Work

2.1 Multi-Task Reinforcement Learning

Multi-task reinforcement learning (MTRL) seeks to improve sample efficiency by sharing knowledge across related tasks, while avoiding negative transfer caused by conflicting objectives (D'Eramo et al., 2020; Teh et al., 2017; Parisotto et al., 2016; Rusu et al., 2016). Existing approaches mainly differ in *what* is shared and *how* task interference is controlled. Gradient-based methods such as GradNorm (Chen et al., 2018), PCGrad (Yu et al., 2020a), and CAGrad (Liu et al., 2021) balance or modify multi-task updates, whereas modular approaches learn task-dependent sharing structures. Soft-Modularization (Yang et al., 2020) routes activations through reusable modules, CARE (Sodhani et al., 2021) combines context-conditioned encoders, and PaCo (Sun et al., 2022) composes task policies from a shared parameter subspace. Most closely related to our representation design, MOORE (Hendawy et al., 2024) applies Gram–Schmidt orthogonalization to a mixture of experts, encouraging diverse shared representations that can be combined in a task-dependent manner. We build on this idea but use the resulting task-conditioned representation jointly for action generation and latent dynamics prediction.

A complementary line studies differences in task learning progress. HyMTRL (He et al., 2026) addresses asynchronous task convergence by moving mastered tasks from reinforcement learning to imitation-based refinement and resetting the shared critic to recover capacity. Our concern arises at an earlier stage: before mastery, a task may participate in optimization much more frequently than it contributes data. Rather than removing mastered tasks from the RL objective, our collection-matched optimizer samples one task per update tick using the same progress-dependent distribution as environment collection, controlling its expected per-task update-to-data ratio.

Primacy-bias work connects repeated replay and high UTD to overfitting early experience and, in MBRL, to the learned world model (Nikishin et al., 2022; Qiao et al., 2024). Discovery deadlock refers here to a task-level pattern in which weakened exploration signals co-occur with limited task success under a collection–update mismatch.

2.2 World Models for Multi-Task Control

World models support decision making by predicting future states or latent representations. Dreamer-style agents learn from imagined latent trajectories (Hafner et al., 2020; 2025), with the original Dreamer obtaining analytic behavior gradients through its model (Hafner et al., 2020). TD-MPC2 (Hansen et al., 2024) instead plans with shared latent dynamics, while MBVE and STEVE (Feinberg et al., 2018; Buckman et al., 2018) construct value targets from short predicted returns and bootstrapping.

Policy optimization through differentiable dynamics is established. SVG jointly learns models, values, and policies through stochastic value gradients (Heess et al., 2015), while MAAC propagates pathwise derivatives through several model steps and a terminal value (Clavera et al., 2020). Closest in MTRL, PWM optimizes policies through an offline-pretrained world model (Georgiev et al., 2025). Our actor-side VE instead co-trains the model from online task-wise data and freezes it along the actor gradient route.

Recent work scales world models to multi-task settings. MoW (Zhang et al., 2026) introduces modular latent representations and task-conditioned dynamics components to accommodate heterogeneous observations and dynamics across tasks. Our focus is complementary to this architectural scaling problem. We retain a real-data SAC actor–critic as the primary learner and use a short differentiable model-based return as an additional actor objective, with the world-model parameters frozen along this gradient path. More importantly, we study an orthogonal online-learning issue: how frequently each task should update its predictive and control components relative to the rate at which that task acquires new experience.

2.3 World–Action Coupling

World Action Models (WAMs) describe an emerging class of embodied models that couple future prediction with action generation rather than treating prediction as an isolated auxiliary task (Wang

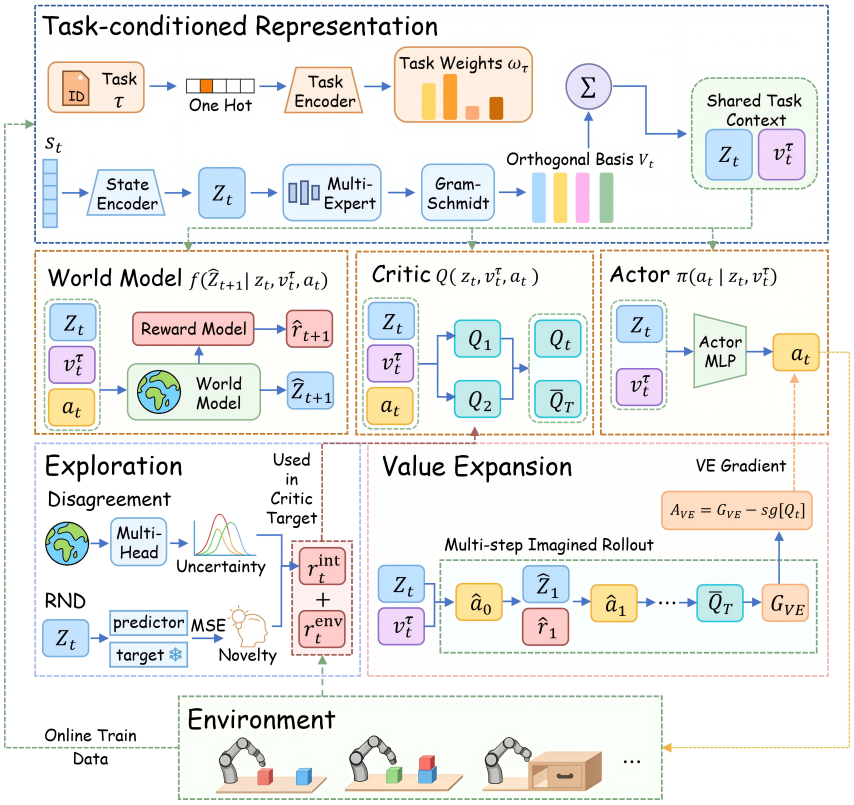

Figure 2: **WAM-MTRL-VE.** A shared task-conditioned context feeds the actor, critic, and latent world model. Imagined futures produce an actor-side VE gradient. Disagreement and RND support exploration.

et al., 2026). Existing work spans cascaded designs, in which predicted futures condition subsequent action generation, and joint designs that learn prediction and action within a shared representational system. Representative systems typically employ large autoregressive or diffusion backbones and are trained primarily from offline visual and language-conditioned data (Du et al., 2023; Wu et al., 2024; Cheang et al., 2024; Guo et al., 2024; Zhu et al., 2025).

We adopt the *prediction–action coupling principle* rather than the large-scale generative formulation of these systems. Our setting is purely online, state-based MTRL: the policy and latent dynamics model operate on the same task-conditioned context, and predicted futures influence policy learning through a differentiable value-expansion objective. This coupling is combined with collection-matched optimization, yielding a setting in which prediction, action, exploration, and per-task data acquisition co-evolve from scratch.

3 Method

We consider tasks $\mathcal{T} = \{1, \ldots, N\}$ that share state and action spaces and discount γ, but differ in dynamics and rewards. A one-hot descriptor e_τ conditions a shared policy $\pi_\theta(a \mid s, e_\tau)$, and each task maintains an online replay buffer $\mathcal{B}_\tau$. On this setting, we build a purely online MTRL agent on SAC (Haarnoja et al., 2018), without demonstrations, offline pretraining, or planning at deployment. Its two core mechanisms route predicted latent futures into the actor through differentiable value expansion and sample task updates to match the task-wise collection distribution. Figure 2 summarizes the complete architecture, and Algorithm 1 gives the full online training loop.

3.1 SHARED-CONTEXT LATENT WAM COUPLING

For state s_t and task identity τ, an encoder produces $z_t = \mathrm{enc}_\phi(s_t)$. Following MOORE (Hendawy et al., 2024), K_e experts produce $U_t = [u_{t,1}, \ldots, u_{t,K_e}]$, which Gram–Schmidt maps to an orthonormal basis V_t. An unconstrained task encoder supplies mixing coefficients ω_τ, yielding the state- and task-conditioned feature $v_t^\tau = V_t \omega_\tau$. The basis is shared, while ω_τ controls task-specific routing. The policy and latent dynamics use separate trunks but read the same context (z_t, v_t^τ). Their causal, action-conditioned factorization is

$$\hat{p}_{\theta,\psi}(a_t, z_{t+1} \mid z_t, \tau) = \pi_\theta(a_t \mid z_t, v_t^\tau)\, \hat{p}_\psi(z_{t+1} \mid z_t, v_t^\tau, a_t). \tag{1}$$

The term "shared-context coupling" refers to both branches operating on the same task-conditioned construction and to the predicted future entering the actor objective in Section 3.2. The imagined actor rollout starts from the online context, treated as constant by the actor loss, whereas the critic and behavior policy use EMA-encoded contexts. Appendix B specifies this routing.

World-model learning. Let $h_{d,t} = f_d(z_t, v_t^\tau)$ be the dynamics-trunk feature. Each of $K_h = 3$ heads predicts $\hat{z}_{t+1}^{(k)} = g_k(h_{d,t}, a_t)$, and their mean $\bar{z}_{t+1} = K_h^{-1} \sum_k \hat{z}_{t+1}^{(k)}$ feeds the reward predictor. We use an EMA encoder to form the current and next-state targets $\tilde{z}_t = \mathrm{enc}_{\bar{\phi}}(s_t)$ and $\tilde{z}_{t+1} = \mathrm{enc}_{\bar{\phi}}(s_{t+1})$. The complete world-model objective, with $\tilde{r}_t^{\mathrm{env}}$ denoting the normalized and clipped environment reward, is

$$\begin{aligned}\mathcal{L}_{\mathrm{WM}} = \frac{\lambda_{\mathrm{dyn}}}{K_h} \sum_{k=1}^{K_h} \left\| \hat{z}_{t+1}^{(k)} - \tilde{z}_{t+1} \right\|_2^2 + \lambda_{\mathrm{rew}} \left\| r_\psi(\mathrm{sg}[\bar{z}_{t+1}], v_t^\tau) - \tilde{r}_t^{\mathrm{env}} \right\|_2^2 \\ + \lambda_{\mathrm{sig}} \mathcal{L}_{\mathrm{SIGReg}}(z_t) + \lambda_{\mathrm{rnd}} \mathcal{L}_{\mathrm{RND}}(\tilde{z}_t).\end{aligned} \tag{2}$$

The reward stop-gradient prevents reward error from distorting the dynamics readout. SIGReg (Balestriero & LeCun, 2025) prevents latent collapse, while the RND term trains the novelty predictor used in Section 3.3. Every term in Equation 2 is trained from ordinary online replay. Additional success-buffer samples never enter the world-model update. Prediction is therefore action-facing rather than merely auxiliary, because the learned dynamics directly defines the actor objective below.

3.2 ACTOR-SIDE VALUE EXPANSION

Actor-side VE combines the short bootstrapped return of MBVE (Feinberg et al., 2018; Buckman et al., 2018) with pathwise gradients through learned dynamics (Heess et al., 2015; Clavera et al., 2020; Hafner et al., 2020; Georgiev et al., 2025). We use it as an auxiliary actor objective and freeze every model-side parameter. Starting from $(\hat{z}_t, \hat{v}_t^\tau) = (z_t, v_t^\tau)$ and the SAC action $\tilde{a}_t$, the ensemble mean predicts $\hat{z}_{t+1}$ and recursively rolls forward. At imagined step $h \geq 1$, the policy samples $\tilde{a}_{t+h}$ from $(\hat{z}_{t+h}, \hat{v}_{t+h}^\tau)$, where $\hat{v}_{t+h}^\tau$ is obtained by applying the frozen expert mixture to $\hat{z}_{t+h}$. For each rollout transition $h = 0, \ldots, H-1$, an EMA head predicts $\hat{r}_{t+h+1}^{\mathrm{env}}$ from the source context $\hat{v}_{t+h}^\tau$. The resulting return is

$$\begin{aligned}G_{\mathrm{VE},t}^{(H)} = \hat{r}_{t+1}^{\mathrm{env}} + \sum_{h=1}^{H-1} \gamma^h \left[\hat{r}_{t+h+1}^{\mathrm{env}} - \alpha_\tau \log \pi_\theta(\tilde{a}_{t+h} \mid \hat{z}_{t+h}, \hat{v}_{t+h}^\tau) \right] \\ + \mathrm{sg}\left[\gamma^H \left(\bar{Q}(\hat{z}_{t+H}, \hat{v}_{t+H}^\tau, \tilde{a}_{t+H}) - \alpha_\tau \log \pi_\theta(\tilde{a}_{t+H} \mid \hat{z}_{t+H}, \hat{v}_{t+H}^\tau) \right) \right].\end{aligned} \tag{3}$$

It augments the ordinary SAC actor objective as

$$\mathcal{L}_{\pi,\tau} = \mathbb{E}\Big[\alpha_\tau \log \pi_\theta(\tilde{a}_t \mid z_t, v_t^\tau) - Q(\tilde{z}_t, \tilde{v}_t^\tau, \tilde{a}_t) - \lambda_{\mathrm{VE}} \big(G_{\mathrm{VE},t}^{(H)} - \mathrm{sg}[Q(\tilde{z}_t, \tilde{v}_t^\tau, \tilde{a}_t)] \big) \Big]. \tag{4}$$

During the rollout, all model-side parameters are frozen but their input Jacobians remain active. Gradients therefore pass through predicted latents and the action chain into π_θ without updating the world model. Direct VE gradients arise from the attached predicted-reward chain and intermediate entropy terms. The terminal bootstrap and critic baseline are stop-gradient. The terminal term affects the numerical return, not the terminal value gradient, but can alter the gradient indirectly through the mask induced by clipping $G_{\mathrm{VE},t}^{(H)}$ to $[-100, 100]$. Appendix E details alternative routes and gradient-isolation checks.

3.3 World-Model Uncertainty and Intrinsic Exploration

The dynamics ensemble also estimates action-conditioned epistemic uncertainty. For predictions $\{\hat{z}'^{(k)}\}_{k=1}^{K_h}$, we average coordinate-wise variance, $u_{\text{dis}} = d_z^{-1} \sum_j \text{Var}_k[\hat{z}_j'^{(k)}]$ (Pathak et al., 2019). We complement it with RND (Burda et al., 2019): a trainable predictor f_{pred} matches a fixed random target f_{tgt} on the EMA latent, producing $u_{\text{RND}} = d_r^{-1} \| f_{\text{pred}}(\tilde{z}) - f_{\text{tgt}}(\tilde{z}) \|_2^2$. Disagreement is action-conditional and dynamics-aware, whereas RND measures state novelty.

Because raw magnitudes are not comparable across tasks, each channel is divided by a task-specific running standard deviation, denoted by $\sigma_\tau(\cdot)$. Exploration is attenuated by collection progress $g_\tau = \text{SR}_\tau^{\text{collect,EMA}}$:

$$r_t^{\text{int}} = \min((1 - g_\tau)\beta_{\text{dis}}\sigma_\tau(u_{\text{dis}}), c) + \min((1 - g_\tau)\beta_{\text{RND}}\sigma_\tau(u_{\text{RND}}), c)\,. \tag{5}$$

The SAC critic target uses $\tilde{r}_t^{\text{env}} + r_t^{\text{int}}$ for ordinary online samples. The RND predictor and dynamics ensemble are updated only from ordinary online replay, and evaluation success never feeds g_τ. Novelty therefore drives discovery while a task is unsolved and decays as successful behavior appears.

If a slowly collected task is updated at every global tick, both ensemble variance and RND error can be fitted down on repeatedly reused data before new behavior is collected. Collection matching ties their expected training rate to the rate at which the task supplies data. This controls one proposed route to premature signal decay, but schedule comparisons alone do not isolate it from simultaneous actor, critic, and representation-learning effects.

3.4 Collection-Matched Stochastic Optimization

A joint update that trains every task once per k global interactions gives task τ an update-to-data ratio $\text{UTD}_\tau \approx 1/(kp_\tau)$, where p_τ is its share of collected data. Under uniform collection this is N/k, rather than the single-task ratio $1/k$, repeatedly fitting slowly replenished replay.

We instead draw *one* task per optimization tick. Let $w_\tau > 0$ be its progress-dependent collection weight and $\mathcal{E}_t = \{\tau : |\mathcal{B}_\tau| \geq B\}$ the replay-ready set. Collection and update probabilities are

$$p_t(\tau) = \frac{w_\tau}{\sum_j w_j}, \qquad q_t(\tau) = \frac{w_\tau \mathbf{1}\{\tau \in \mathcal{E}_t\}}{\sum_{j \in \mathcal{E}_t} w_j}, \qquad \text{UTD}_\tau \approx \frac{q_\tau}{kp_\tau}. \tag{6}$$

Once all buffers are ready, $q_t = p_t$ and $\text{UTD}_\tau = 1/k$, independent of task count and unequal collection shares. This is expectation matching, not a hard freshness constraint. We set w_τ toward $(1 - g_\tau) + \epsilon$, so difficult tasks receive more data and proportionally more updates.

3.5 Capacity-Gated Consolidation

Intrinsic exploration can discover a rare success without ensuring that it is retained. We therefore keep a per-task success buffer $\mathcal{B}_\tau^+$ (Oh et al., 2018). Its mixing ratio is gated by buffer occupancy rather than success rate, so the first complete successful trajectory opens a small consolidation channel once the buffer contains $n_{\min}$ transitions. Extra success samples train the critic and the complete actor objective, including both SAC and actor-side VE. They receive no immediate intrinsic bonus and never enter world-model or RND losses. Thus discovery remains driven by ordinary online replay, while consolidation is value-based rather than a separate behavior-cloning phase. Mastered tasks are not removed from training. The precise occupancy-to-mixing schedule is given in Appendix C.

4 Experimental Results

4.1 Experimental Setup

We evaluate on the random-goal Meta-World MT10 and MT50 benchmarks (Yu et al., 2020b). Training is purely online and uses 1M global environment interactions on MT10. Every 10^4 interactions, we evaluate the deterministic policy for 10 episodes per task and average success first

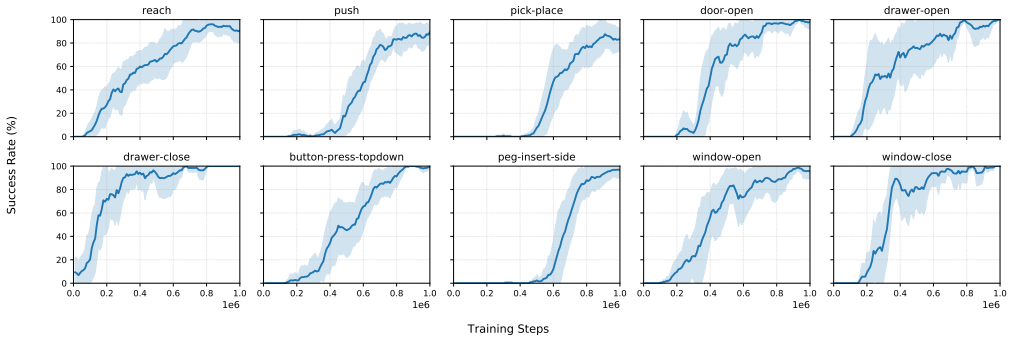

Figure 3: **Per-task MT10 learning curves.** Each panel reports the mean success over five seeds. Shading denotes ± 1 standard deviation. Curves use a centered five-evaluation moving average for visualization. Each run uses only 1M total environment interactions.

Table 1: **Meta-World MT50-rand comparison.** Success is averaged across tasks. "Steps" denotes total environment interactions across all 50 tasks, and H is the maximum episode length. Baseline values and budgets follow the comparison protocols reported in the cited papers. † evaluates the selected checkpoint from one training run on five independent evaluation seeds with 10 episodes per task and seed. Its deviation is across evaluation seeds, not independently trained agents.

Method	Obs.	H	Training steps ↓	Success (%) ↑
MTSAC	State	150	20M	49.3 ± 1.5
CARE	State	150	20M	50.8 ± 1.0
PaCo	State	150	20M	57.3 ± 1.3
MOORE	State	150	100M	72.9 ± 3.3
MoW	Image	500	15M	74.5 ± 1.1
MOORE + HyMTRL	State	200	75M	$\mathbf{86.2 \pm 3.4}$
WAM-MTRL-VE	State	150	**12**M	$82.1 \pm 0.5^{\dagger}$

over episodes and tasks, then over five training seeds. MT10 curves show the mean and one standard deviation across training seeds. A centered five-evaluation moving average is used only for visualization, and all scalar statistics use raw evaluation records. For MT50, each variant selects the best checkpoint according to its periodic evaluation during one training run. We then reevaluate that checkpoint on five independently seeded random-goal suites, with 10 episodes per task and evaluation seed. The reported deviation therefore measures evaluation variation rather than training-seed variation. Each run uses one NVIDIA RTX 3090 GPU with 24 GB memory. Full implementation, evaluation, and hyperparameter details appear in Appendix F.

4.2 Main Results

MT10. Figure 3 shows that WAM-MTRL-VE learns all ten tasks within 1×10^6 global interactions despite substantial variation in task discovery times. This corresponds to only 0.1M interactions per task on average. Across five seeds, the mean of the per-run best observed success rates is $98.0\% \pm 1.9\%$. Because this statistic selects a different checkpoint per seed, we also report the fixed-budget result, $96.2\% \pm 1.6\%$ at 1M steps, for checkpoint-aligned comparison.

MT50. Table 1 compares WAM-MTRL-VE with reported results from MTSAC, CARE, PaCo, MOORE, MoW, and MOORE with HyMTRL (Yu et al., 2020b; Sodhani et al., 2021; Sun et al., 2022; Hendawy et al., 2024; Zhang et al., 2026; He et al., 2026). The selected checkpoint from the 12M-interaction run obtains $82.1 \pm 0.5\%$ across five independently seeded evaluation suites, covering 2,500 episodes in total. The result exceeds the reported MOORE and MoW means with fewer interactions. MOORE with HyMTRL reports a higher mean using 75M interactions and a longer episode horizon.

Table 2: **MT10 task-update scheduling ablation.** Values are the mean ± standard deviation over three training seeds at 1M interactions. The bottom two rows complete the same-seed schedule–intrinsic factorial. Naive joint processes roughly N times as many task minibatches per update tick as the single-task variants.

Update schedule	Intrinsic	SR@1M (%) ↑	AUC (%) ↑
Collection-matched (ours)	On	$\mathbf{97.3 \pm 0.6}$	53.8 ± 1.5
Uniform random-single	On	93.3 ± 4.9	51.6 ± 3.4
Freshness-gated	On	52.0 ± 5.3	25.7 ± 0.9
Naive joint	On	74.7 ± 1.5	$\mathbf{56.2 \pm 2.3}$
Collection-matched	Off	92.0 ± 1.0	55.3 ± 3.2
Naive joint	Off	69.7 ± 4.9	52.8 ± 9.1

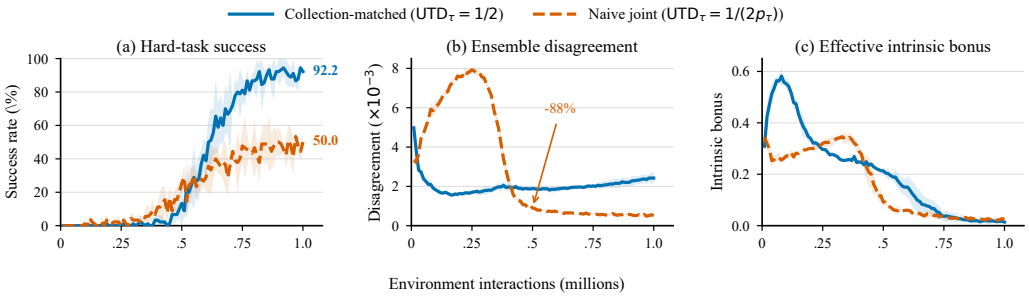

Figure 4: **Discovery-deadlock diagnostics on MT10.** We select the three tasks with the latest median evaluated discovery time under collection-matched optimization (SR$\geq$ 0.1): Pick-Place, Peg-Insert-Side, and Push. We first average these tasks within each run and then report the mean and one standard deviation over three common training seeds. Raw disagreement and the effective intrinsic bonus are training-time exponential moving averages. The latter combines task-normalized disagreement and RND after progress gating, so it does not isolate raw RND, normalization, or gating effects. UTD values in the legend are schedule-implied expectations rather than empirical counters.

4.3 Ablation Studies

Task-update scheduling. We first isolate the two scheduling failures identified in Section 3.4: an inflated task-wise update-to-data ratio (UTD) and temporally concentrated task gradients. All variants use the same demand-weighted collection rule, replay buffers, losses, and one-million-interaction MT10 budget. Our *collection-matched* schedule draws one replay-ready task at each update tick with $q_\tau = p_\tau$. *Uniform random-single* retains one task batch per tick but sets $q_\tau = 1/N$, isolating distribution matching. *Freshness-gated* updates only tasks that have accumulated $\kappa = 2$ new transitions. It preserves the nominal per-task UTD, but episode-wise collection organizes its updates into long task-local blocks. Finally, *naive joint* updates every replay-ready task at each tick, processing approximately N task batches and inflating task τ's UTD from $1/2$ to $1/(2p_\tau)$, which equals $N/2$ under uniform collection. Table 2 reports the mean over common training seeds 42, 100, and 2026. In the intrinsic-off variants, $\beta_{\text{dis}} = \beta_{\text{RND}} = 0$, while the world-model and RND training objectives and all remaining update rules are unchanged. AUC is the normalized area under the raw average-success curve over $[0, 1\text{M}]$. Reported differences are computed from unrounded means.

Collection matching gives the highest fixed-budget mean at $97.3\% \pm 0.6\%$. Uniform random-single uses the same one-task-per-tick compute but reaches $93.3\% \pm 4.9\%$. Thus matching updates to the nonuniform collection distribution raises the observed mean endpoint and reduces its observed standard deviation in these runs. Freshness gating performs substantially worse at $52.0\% \pm 5.3\%$. Although it controls UTD, consecutive updates follow the task currently being collected and expose shared parameters to long blocks of correlated task-specific gradients. Matching the update rate alone is therefore insufficient in this implementation. Updates also need to remain interleaved across tasks.

Naive joint uses substantially more task-batch compute and achieves the largest AUC, $56.2\% \pm 2.3\%$, but a substantially lower fixed-budget endpoint of $74.7\% \pm 1.5\%$. Figure 4 examines the associated

learning dynamics on the three latest-discovered tasks. Under naive joint, ensemble disagreement decreases by 88% between 0.3M and 0.5M interactions, while mean hard-task success remains only 25.6% at 0.5M. The effective intrinsic bonus is then 0.092, less than half the 0.213 retained by collection matching. At 1M interactions, collection matching reaches 92.2% hard-task success, whereas naive joint reaches 50.0%. These trends show that exploration-signal attenuation and poor hard-task performance co-occur, but do not establish mediation. The bottom two rows of Table 2 provide a same-seed 2×2 test. Removing intrinsic rewards reduces the endpoint by 5.3 points under collection matching and by 5.0 points under naive joint. Across the three paired seeds, the estimated endpoint interaction is 0.3 percentage points, with a sample standard deviation of 4.5 points. The scheduling advantage persists without intrinsic rewards and likely also reflects broader actor, critic, and representation-learning effects.

Value expansion and consolidation. Table 3 examines the remaining components at the same 1M-step MT10 budget and reports best-checkpoint evaluations on MT50. The MT10 default, first-step, no-VE, and no-success-replay variants all use training seeds 42, 100, and 2026.

Table 3: **Component ablations.** MT10 uses raw evaluations through 1M interactions and reports mean $\pm$ standard deviation over three training seeds for every variant. Each MT50 entry evaluates the best checkpoint from one training run on five independently seeded goal suites. Its deviation is therefore across evaluation seeds, not training seeds.

Variant	MT10 SR (%) ↑	MT10 AUC (%) ↑	MT50 SR (%) ↑
Multi-step VE (default)	97.3 ± 0.6	53.8 ± 1.5	82.1 ± 0.5
First-step VE	92.7 ± 4.9	50.0 ± 6.2	79.2 ± 1.4
No VE	94.7 ± 2.1	50.6 ± 3.9	77.9 ± 0.7
No success replay	88.7 ± 7.5	45.3 ± 4.1	77.6 ± 2.0

The default multi-step route improves the three-seed endpoint by 2.7 percentage points over no VE and by 4.7 points over first-step VE, although the AUC uncertainty indicates that MT10 only weakly separates the three routes. Removing success replay reduces the endpoint from $97.3\% \pm 0.6\%$ to $88.7\% \pm 7.5\%$ and reduces AUC from $53.8\% \pm 1.5\%$ to $45.3\% \pm 4.1\%$, supporting its role in policy consolidation. On MT50, the default best checkpoint reaches $82.1\% \pm 0.5\%$, compared with $79.2\% \pm 1.4\%$ for first-step VE, $77.9\% \pm 0.7\%$ without VE, and $77.6\% \pm 2.0\%$ without success replay. The default therefore leads these variants by 2.9–4.5 percentage points. Each MT50 number comes from one trained checkpoint, however, and its deviation only captures variation across evaluation suites. These results provide directional component evidence rather than training-seed uncertainty or controlled effect sizes across independently trained agents.

5 Conclusion

WAM-MTRL-VE couples latent prediction to action learning and collection-matched task optimization. A frozen-model, short-horizon return shapes the actor, while matched updates retain the single-task UTD. The same-seed factorial shows that the scheduling advantage persists without intrinsic rewards, pointing to broader optimization effects rather than intrinsic-signal preservation alone. Limitations are model bias and reliance on replay-ready buffers, matched collection shares, known task identities, and success signals.

References

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the International Conference on Machine Learning*, pp. 794–803, 2018.

Ignasi Clavera, Yao Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations*, 2020.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems*, 2023.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Ignat Georgiev, Varun Giridhar, Nicklas Hansen, and Animesh Garg. PWM: Policy learning with multi-task world models. In *International Conference on Learning Representations*, 2025.

Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. In *Advances in Neural Information Processing Systems*, 2024.

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 International Conference on Machine Learning*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Jinmin He, Kai Li, Xiaoyi Dong, Yifan Zang, Yuheng Jing, Yifan Zhang, Junliang Xing, and Jian Cheng. HyMTRL: A hybrid multi-task reinforcement learning framework via phased policy evolution. In *Proceedings of the International Conference on Machine Learning*, 2026.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, 2015.

Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *International Conference on Learning Representations*, 2024.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, 2021.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 16828–16847, 2022.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *International Conference on Learning Representations*, 2016.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *Proceedings of the International Conference on Machine Learning*, 2019.

Zhongjian Qiao, Jiafei Lyu, and Xiu Li. Mind the model, not the agent: The primacy bias in model-based RL. In *European Conference on Artificial Intelligence*, pp. 1824–1831, 2024.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations*, 2016.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, 2021.

Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. PaCo: Parameter-compositional multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, Mike Zheng Shou, Xuanjing Huang, Xipeng Qiu, and Yu-Gang Jiang. World action models: The next frontier in embodied AI. *arXiv preprint arXiv:2605.12090*, 2026.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, 2020.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

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*, pp. 1094–1100, 2020.

Boxuan Zhang, Weipu Zhang, Zhaohan Feng, Wei Xiao, Jian Sun, Jie Chen, and Gang Wang. Mixture-of-world models: Scaling multi-task reinforcement learning with modular latent dynamics. In *International Conference on Learning Representations*, 2026.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Robotics: Science and Systems*, 2025.

A Online Training Algorithm

Algorithm 1 summarizes the interaction, replay routing, and optimization order of the declared collection-matched configuration. The collection task τ_c remains fixed within an episode, whereas the update task τ_u is sampled independently at every optimization tick.

Algorithm 1 WAM-MTRL-VE with collection-matched optimization

Require: Tasks $\mathcal{T}$, update period k, online buffers $\{\mathcal{B}_\tau\}$, success buffers $\{\mathcal{B}_\tau^+\}$
Initialize actor, critic, world model, RND networks, and $g_\tau \leftarrow 0$, $w_\tau \leftarrow 1$ for every τ
Sample an initial collection task τ_c
for global interaction $t = 1, 2, \ldots$ **do**
  Interact with task τ_c and append the transition to $\mathcal{B}_{\tau_c}$ and the current episode record
  if the episode ends **then**
    if the episode succeeds **then**
      Copy its transitions to $\mathcal{B}_{\tau_c}^+$
    end if
    Update the collection-success EMA g_{τ_c} and weight w_{τ_c}
    Sample the next collection task $\tau_c \sim p_t(\tau) \propto w_\tau$
  end if
  if t is an optimization tick and learning has started **then**
    Form $\mathcal{E}_t = \{\tau : |\mathcal{B}_\tau| \geq B\}$
    if $\mathcal{E}_t \neq \varnothing$ **then**
      Sample one update task $\tau_u \sim q_t(\tau) \propto w_\tau \mathbf{1}\{\tau \in \mathcal{E}_t\}$
      Sample pure-online $\mathcal{D}_{\mathrm{WM}}$ from $\mathcal{B}_{\tau_u}$
      Sample mixed $\mathcal{D}_{\mathrm{RL}}$ from $\mathcal{B}_{\tau_u}$ and $\mathcal{B}_{\tau_u}^+$ using ρ_{τ_u}
      Compute intrinsic rewards only for ordinary online rows of $\mathcal{D}_{\mathrm{RL}}$
      Update the critic with the clipped target in Equation 7
      Compute $\mathcal{L}_{\mathrm{WM}}$ on $\mathcal{D}_{\mathrm{WM}}$
      Compute SAC and actor-side VE on $\mathcal{D}_{\mathrm{RL}}$, freezing model-side parameters along the VE path
      Update the joint world-model–actor optimizer with $\mathcal{L}_{\mathrm{WM}} + \mathcal{L}_{\pi,\tau_u}$
      Update the task temperature, target critic, EMA encoder, and EMA reward head
    end if
  end if
end for

B Additional Optimization Details

Online and EMA contexts. The online encoder gives $z_t = \mathrm{enc}_\phi(s_t)$ and the EMA encoder gives $\tilde{z}_t = \mathrm{enc}_{\bar{\phi}}(s_t)$. Applying the same expert mixture to these latents produces v_t^τ and $\tilde{v}_t^\tau$, respectively. The world-model loss and imagined actor rollout start from the online context (z_t, v_t^τ). This context is computed without actor-side gradient tracking, so the actor loss does not train the encoder or experts. The critic uses EMA contexts for both current and next states, while environment interaction and evaluation also query the policy with the EMA context. Thus the actor proposes the training action from the online context in Equation 4, and the critic evaluates it using $(\tilde{z}_t, \tilde{v}_t^\tau)$.

Progress-gated exploration. Let u denote ensemble variance and u_{RND} the RND error. Equation 5 normalizes each with per-task running statistics and gates both by collection-success EMA g_τ. Online critic targets use $\tilde{r}_t^{\mathrm{env}} + r_t^{\mathrm{int}}$. Additional success-buffer rows receive zero immediate bonus. Evaluation success is reserved for reporting.

Clipped soft critic target. Let r_t^{env} be the raw environment reward and $\tilde{r}_t^{\mathrm{env}} = \mathrm{clip}(\mathcal{N}_\tau(r_t^{\mathrm{env}}), -10, 10)$ its value after re-standardization with the current per-task statistics. Let d_t indicate a true termination, excluding time-limit truncation, and let $\bar{Q} = \min(\bar{Q}_1, \bar{Q}_2)$. For $a_{t+1} \sim \pi_\theta(\cdot \mid \tilde{z}_{t+1}, \tilde{v}_{t+1}^\tau)$, the implemented target is

$$y_t = \tilde{r}_t^{\mathrm{env}} + r_t^{\mathrm{int}} + \gamma(1 - d_t)\big[\bar{Q}(\tilde{z}_{t+1}, \tilde{v}_{t+1}^\tau, a_{t+1}) - \alpha_\tau \log \pi_\theta(a_{t+1} \mid \tilde{z}_{t+1}, \tilde{v}_{t+1}^\tau)\big], \tag{7}$$

where r_t^{int} is given by Equation 5 for ordinary online rows and is set to zero for rows sampled from the success buffer. The twin critics minimize the sum of squared errors to y_t using the EMA-encoded current context $(\tilde{z}_t, \tilde{v}_t^\tau)$.

Optimizer partition. For the task $\tau_t \sim q_t$ selected at a tick, the implementation uses three objectives and three Adam optimizers:

$$\mathcal{L}_{Q,t} = \mathcal{L}_{\mathrm{critic},\tau_t}, \qquad \mathcal{L}_{\mathrm{joint},t} = \mathcal{L}_{\mathrm{WM},\tau_t} + \mathcal{L}_{\pi,\tau_t}, \qquad \mathcal{L}_{\alpha,t} = \mathcal{L}_{\alpha,\tau_t}. \tag{8}$$

The joint objective draws a pure-online world-model batch and a separate RL batch that may include capacity-gated success replay.

C Capacity-Gated Success Replay Details

Each task τ keeps, alongside its online replay $\mathcal{B}_\tau$, a *success buffer* $\mathcal{B}_\tau^+$ that stores every transition of any episode the environment flags as successful. The two sources serve different losses. The critic and complete actor update draw a mixed minibatch with on average $\rho_\tau B$ samples from $\mathcal{B}_\tau^+$, while the world model of Equation 2 trains on $\mathcal{B}_\tau$ alone. Gating the ratio on the online success rate, $\rho_\tau \propto \mathrm{SR}_\tau$, creates a bootstrapping deadlock on hard tasks (zero success rate forces $\rho_\tau = 0$), so we gate on *buffer occupancy* $n_\tau = |\mathcal{B}_\tau^+|$:

$$\rho_\tau \;=\; \begin{cases} 0 & \text{if } n_\tau < n_{\min}, \\ \rho_{\max} \cdot \min\!\big(1,\; n_\tau / n_{\mathrm{warm}}\big) & \text{if } n_\tau \geq n_{\min}, \end{cases} \tag{9}$$

with $n_{\min}$ roughly one successful episode and n_{warm} the occupancy at which the ratio saturates to $\rho_{\max}$. The moment $\mathcal{B}_\tau^+$ crosses $n_{\min}$ the channel opens, and because $\rho_\tau \leq \rho_{\max}$ successful transitions remain a minority of the batch, preserving exploratory pressure. Replayed success transitions serve consolidation rather than exploration. Their rewards are re-standardized with the *current* per-task statistics at train time rather than the stale collection-time ones, and their immediate reward carries no intrinsic bonus—the bootstrap critic is still trained with intrinsic-augmented targets elsewhere, so the target is not purely extrinsic. Unlike HyMTRL (He et al., 2026), mastered tasks are never removed from Equation 8, and successful transitions fold into the same critic and actor objectives rather than a separate behavior-cloning pass. The actor receives both SAC and VE gradients on the success-biased batch, while the critic regresses the same transitions, so consolidation remains value-driven rather than cloning-driven.

D SIGReg Objective

The Sketched Isotropic Gaussian Regularization (SIGReg) of LeJEPA (Balestriero & LeCun, 2025) draws P unit-norm sketch directions $A \in \mathbb{R}^{d_z \times P}$ and matches the empirical characteristic function of the projected latents to that of an isotropic Gaussian on a uniform grid $t_k \in [0, 3]$. For a minibatch of size n, the implementation uses Gaussian-windowed trapezoidal weights $\omega_k = c_k e^{-t_k^2/2}$, where $c_k = \Delta t$ at the endpoints and $c_k = 2\Delta t$ in the interior:

$$\mathcal{L}_{\mathrm{SIGReg}}(z) \;=\; n\,\mathbb{E}_A\!\left[\sum_{k=1}^{K} \omega_k \Big(\big|\mathbb{E}_z[\cos(t_k\, A^\top z)] - e^{-t_k^2/2}\big|^2 \;+\; \big|\mathbb{E}_z[\sin(t_k\, A^\top z)]\big|^2 \Big)\right]. \tag{10}$$

Because the standard normal is uniquely characterized by $\varphi(t) = e^{-t^2/2}$, matching both cos- and sin-moments drives the sketched distribution to $\mathcal{N}(0, I)$. The encoder is shaped by distributional regularization without any variance/whitening surrogate.

E Value-Expansion Variants and Gradient Verification

We compare three routings of the VE term in Section 4. **No VE** sets $\lambda_{\mathrm{VE}} = 0$. The default **multi-step reward chain** keeps every rollout step up to $H-1$ attached while freezing all model-side parameters, including the dynamics trunk, ensemble heads, action embedder, and orthogonal experts. The gradient therefore flows through the action chain and predicted latents into the policy only, while the terminal bootstrap remains detached and contributes only to the numerical return and its clipping mask. The conservative **first-step VE** variant allows only the first predicted reward $\hat{r}^{\mathrm{env}}_{t+1}(\tilde{a}_t)$ to carry gradient and evaluates the $H-1$ tail under disabled gradient tracking.

Two verification measurements accompany these variants. First, in the multi-step mode the dynamics and expert parameters receive zero gradient from the actor loss (up to floating-point noise), confirming that freezing parameters—not detaching inputs—is the correct shield. Detaching a module's input would block the very chain the gradient is meant to traverse. Second, for a fixed initial state and first-action sample, the first-step return before clipping has the form

$$G_{H,t}(\theta) = \hat{r}^{\mathrm{env}}_{t+1}(\theta) + \mathrm{sg}[C_{H,t}], \qquad \nabla_\theta G_{H,t} = \nabla_\theta \hat{r}^{\mathrm{env}}_{t+1},$$

so its gradient is horizon-independent before clipping. The implementation, however, clips the complete return to $[-100, 100]$. Almost everywhere, this operation preserves the gradient above for

unsaturated returns and suppresses it for saturated returns. The detached tail $C_{H,t}$, including the terminal bootstrap, can therefore introduce horizon dependence by changing whether the complete return saturates, even though the tail supplies no direct gradient. The experiments compare the acquisition and stability trade-offs of these routes. The declared final configuration is the multi-step reward chain with a detached terminal bootstrap.

F Training Details

Optimizers. Equation 8 is partitioned across *three Adam optimizers* and *three backward passes* for the selected task τ_t. The critic optimizer minimizes $\mathcal{L}_{\mathrm{critic},\tau_t}$. The joint world-model–actor optimizer minimizes $\mathcal{L}_{\mathrm{WM},\tau_t} + \mathcal{L}_{\pi,\tau_t}$ with one combined backward pass. The two losses touch disjoint parameter groups. The actor loss reaches only the policy trunk and heads because its encoder and expert inputs are treated as constants, whereas the world-model losses reach the encoder, experts, dynamics trunk, and auxiliary networks. Gradients are clipped jointly across this union. The temperature optimizer minimizes $\mathcal{L}_{\alpha,\tau_t}$.

Parameter groups and learning rates. The joint optimizer uses two parameter groups: the policy trunk f_π and policy heads use the base learning rate η, while the shared encoder, the orthogonal experts, the dynamics trunk f_d, the action embedder, the K_h dynamics heads, the reward predictor, and the RND predictor use a reduced rate $0.3\,\eta$—the asymmetry slows the non-stationary components the actor depends on, stabilizing the representation the actor reads from.

Target networks. The EMA target encoder $\mathrm{enc}_{\bar{\phi}}$, the EMA reward predictor $\bar{r}_\psi$ used in VE rollouts, and the target critic $\bar{Q}$ are updated only by Polyak averaging, never by gradient descent. The RND target network is frozen at initialization and never updated.

Gradient routing. In the default (multi-step) VE mode, every rollout step up to $H-1$ is evaluated with gradients enabled while the action embedder, the dynamics heads, the dynamics trunk, and the orthogonal experts are temporarily frozen, so the gradient flows through the action chain and the predicted latents into the policy only. The terminal bootstrap is computed with gradient tracking disabled and has a numerical, but not direct gradient, role. A conservative first-step VE variant attaches only the first predicted reward and computes the whole $H-1$ tail without gradient tracking. The complete VE value is clipped to ± 100 before entering the actor loss, so detached terms can still alter the clipping mask. Joint gradient clipping (max norm 1.0) is applied across all parameters of the joint world-model–actor optimizer.

Two-batch data routing. Each update draws two batches for the selected task τ_t. The RL minibatch has size B, mixes online transitions with success-buffer transitions at ratio ρ_τ from Equation 9, and feeds the critic loss and the complete actor objective, including SAC and VE. The pure-online world-model minibatch has size B_{wm} and feeds the dynamics, reward, SIGReg, and RND-predictor losses. Success-buffer samples therefore never shape the world model. Successful interactions still occur naturally in ordinary online replay. All buffers store raw rewards, and every sampled row—online or success—is re-standardized with the *current* per-task normalizer statistics (clipped to ± 10) at train time, so a transition drawn from either buffer carries the same reward label. The intrinsic bonus of Equation 5 is computed on the online rows of the RL minibatch only.

Collection-matched scheduling. Every $k = 2$ global environment steps, the final configuration samples one replay-ready task from q_t in Equation 6. It does not require a fresh-step counter. Collection and update sampling use independent pseudo-random generators, preventing an update draw from perturbing the subsequent collection sequence. Both use the same positive task weights. Let $y_{\tau,m} \in \{0,1\}$ indicate whether the m-th collected episode of task τ succeeds. At the end of that episode, the collection-success statistic and sampling weight are updated as

$$\begin{aligned} g_\tau &\leftarrow \beta_g g_\tau + (1-\beta_g) y_{\tau,m}, && \beta_g = 0.95, \\ w_\tau &\leftarrow \beta_w w_\tau + (1-\beta_w)\big[(1-g_\tau) + \epsilon\big], && \beta_w = 0.9, \quad \epsilon = 0.05. \end{aligned} \tag{11}$$

Here g_τ is the collection-success EMA. Under the declared collection-driven gating rule, evaluation never refreshes these weights and therefore cannot steer the training schedule. Once all buffers are

ready, collection matching gives the expected $\mathrm{UTD}_\tau = 1/k = 0.5$ in Equation 6. By contrast, updating every task at each tick gives $N/2$ under uniform collection. We use $B = B_{\mathrm{wm}} = 256$. A fresh-counter schedule, uniform random-single updates, and all-task joint updates are comparison modes, not the proposed default. The optional starvation multiplier is set to 1, so it is disabled in the final configuration.

Declared benchmark configurations. For MT10, we use Reach, Push, Pick-Place, Door-Open, Drawer-Open, Drawer-Close, Button-Press-Topdown, Peg-Insert-Side, Window-Open, and Window-Close. The MT10 study uses 1×10^6 total steps, four experts, seeds 42, 66, 100, 585, and 2026, replay capacity 5×10^5, and evaluation every 10^4 steps. MT50 uses the full 50-task list, 1.2×10^7 total steps, six experts, seed 2026, replay capacity 2×10^5, and evaluation every 5×10^4 steps. Both use multi-step VE, collection-driven gating, collection-matched single-task updates, and an update period of two environment steps. Each run is executed on a single NVIDIA RTX 3090 GPU with 24 GB memory.

Timeout handling. Only true terminations cut the bootstrap. Time-limit truncations are stored as regular transitions with γ-discounted targets, which matters for tasks whose successes cluster at the episode horizon.

Hyperparameters. Table 4 summarizes the default settings used in the declared MT10 and MT50 configurations.

Table 4: Default hyperparameters. Where two values are given, they refer to MT10 / MT50 only in rows explicitly labeled by benchmark.

SAC and optimization		WAM and VE	
learning rate η	3×10^{-4}	$(\lambda_{\mathrm{dyn}}, \lambda_{\mathrm{rew}}, \lambda_{\mathrm{sig}}, \lambda_{\mathrm{rnd}})$	$(1,\ 0.5,\ 10^{-3},\ 0.1)$
discount γ	0.99	$(\lambda_{\mathrm{VE}},\ H)$	$(0.1,\ 3)$
target-network update rate	0.005	experts K_e (MT10 / MT50)	4 / 6
EMA decay	0.996	ensemble heads K_h	3
target entropy	$-\lvert\mathcal{A}\rvert$	reward / critic-target clip	± 10 / ± 100
gradient clipping (max norm)	1.0	VE clip	± 100
actor / critic hidden layers	2 / 2	latent / hidden dimensions	256 / 512
		SIGReg knots / projections	17 / 512
Scheduling and buffers		**Exploration and protocol**	
batch B / B_{wm}	256 / 256	$(\beta_{\mathrm{dis}},\ \beta_{\mathrm{RND}})$	$(0.1,\ 0.05)$
replay capacity (MT10 / MT50)	5×10^5 / 2×10^5	bonus cap c	1.0
success-buffer capacity	10^4	$(\rho_{\max},\ n_{\min},\ n_{\mathrm{warm}})$	$(0.3,\ 150,\ 3000)$
tasks per update tick	1	collection floor ϵ	0.05
update period	2 environment steps	learning starts	5000 steps
evaluation frequency (MT10 / MT50)	10^4 / 5×10^4	evaluation episodes	10
total steps (MT10 / MT50)	1M/12M	VE gradient route	multi-step
update-task distribution	collection-matched	terminal VE gradient	detached

G Evaluation and Reporting Protocol

Let $s_{\tau,e}(t) \in \{0, 1\}$ denote success for task τ in evaluation episode e at checkpoint t. We first average over $E = 10$ episodes and all N tasks,

$$\bar{S}(t) = \frac{1}{N} \sum_{\tau=1}^{N} \frac{1}{E} \sum_{e=1}^{E} s_{\tau,e}(t). \tag{12}$$

For MT10, SR@1M is $\bar{S}(10^6)$ and the normalized area under the learning curve is

$$\mathrm{AUC}_T = \frac{1}{T} \int_0^T \bar{S}(t)\, \mathrm{d}t, \qquad T = 10^6. \tag{13}$$

We compute the integral by the trapezoidal rule on unsmoothed evaluation checkpoints and set $\bar{S}(0) = 0$. Reported MT10 means and standard deviations are across independently trained seeds. Moving averages appear only in plots.

For MT50, periodic evaluation during training determines the saved best checkpoint for each variant. We reevaluate that fixed checkpoint on five independently seeded random-goal suites, using 10 episodes per task and suite. Thus each MT50 entry summarizes 2,500 episodes. Its mean and sample standard deviation are across the five evaluation suites for a single training run, not across independently trained seeds. Best-checkpoint MT50 comparisons are therefore descriptive component evidence and do not estimate training-run variance.

H Additional MT10 Statistics

The task curves differ sharply in when learning begins: the drawer and window tasks are acquired early, whereas Pick-Place, Peg-Insert-Side, and Button-Press-Topdown improve later or show larger seed-to-seed variation. Table 5 reports the fixed-budget endpoint rather than selecting a checkpoint separately for each task.

Table 5: MT10 success at the fixed 1M-step endpoint. Values are the unsmoothed mean $\pm$ standard deviation over five seeds.

Task	Final success (%)
reach	96.0 ± 8.9
push	92.0 ± 8.4
pick-place	82.0 ± 4.5
door-open	100.0 ± 0.0
drawer-open	100.0 ± 0.0
drawer-close	100.0 ± 0.0
button-press-topdown	100.0 ± 0.0
peg-insert-side	96.0 ± 8.9
window-open	96.0 ± 5.5
window-close	100.0 ± 0.0
Task average	96.2 ± 1.6

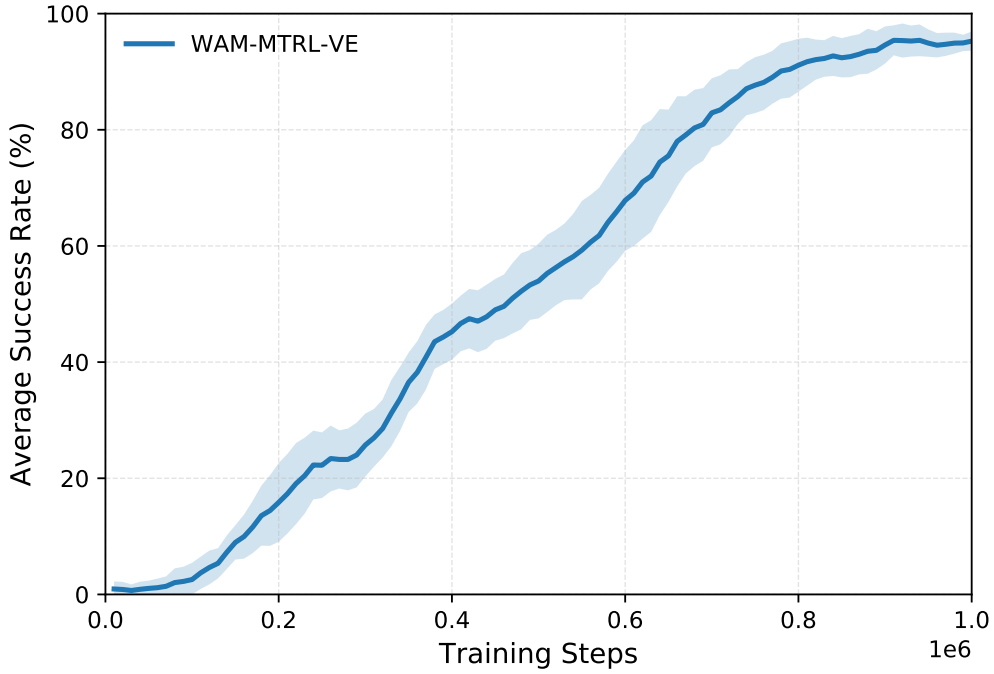

Figure 5: **Aggregate MT10 learning curve.** Mean task success over five seeds. Shading denotes ± 1 standard deviation. Smoothing is used only for visualization. All reported scalar values are unsmoothed.

I MT50 Training Dynamics

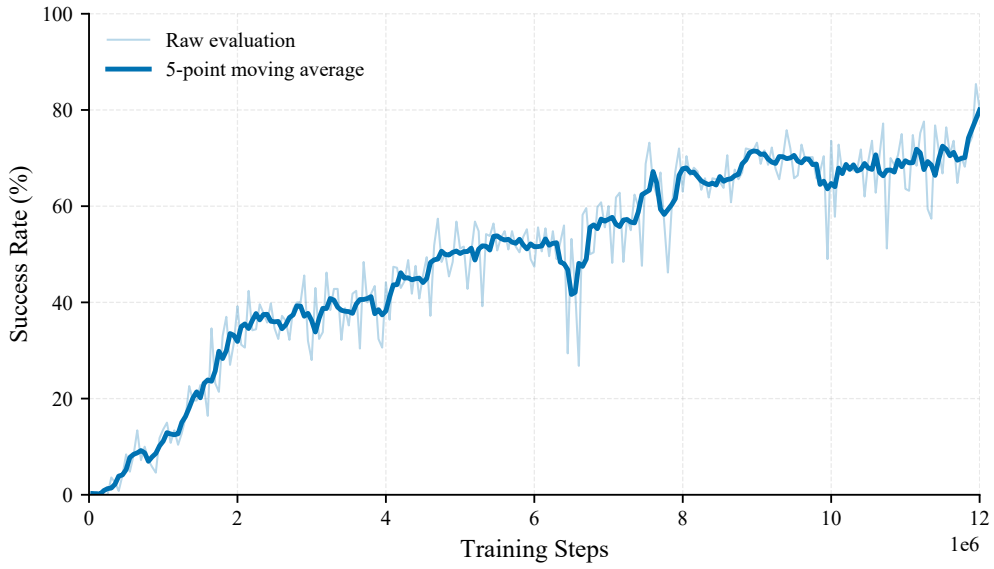

Figure 6: **MT50 training curve.** Periodic average success rate during MT50 training. The light curve shows raw evaluations, while the dark curve shows a centered five-point moving average. Each evaluation is averaged over 10 episodes per task across all 50 tasks.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the training algorithm in Appendix A, implementation details, hyperparameters, and training seeds in Appendix F, and the evaluation and reporting protocol in Appendix G. Upon acceptance, we will publicly release the source code for WAM-MTRL-VE, together with training and evaluation scripts and experiment configurations, to support reproduction of the reported results.

J GENERATIVE AI USE DISCLOSURE

In compliance with the ICLR 2027 policy on LLMs, we declare that LLMs were used only for language polishing. All research ideas, methodology, experiments, data analysis, and the intellectual content of this paper were solely developed by the human authors. The LLM did not contribute to any scientific content. The authors have carefully verified all LLM-assisted revisions to ensure no inaccurate information, hidden instructions, or prompt injections were introduced. This usage fully adheres to the ICLR Code of Ethics.