DEMYSTIFYING CROSS-DOMAIN POLICY ADAPTATION WITH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-domain policy adaptation seeks to reuse source-domain knowledge under dynamics shifts with limited target-domain interaction. Model-based methods address this problem by adapting source-pretrained world models, but their apparent sample-efficiency advantage may partly arise from value transfer: model-based methods often retain source-trained value functions, while model-free baselines learn critics from scratch. When we give model-free baselines the same source-trained critic, they initially adapt faster than their Dyna-style counterparts. Further analysis identifies that dynamics ensembles and prior-guided data collection jointly improve Dyna-style adaptation, allowing it to outperform the model-free baseline on average. Guided by these findings, we propose DOMAINADEPT, which improves model-based planning both with a dynamics ensemble to improve its accuracy and with "tethered" proposal policy fine-tuning to limit policy drift during data collection. Across four contact-rich manipulation tasks and one locomotion task, DOMAINADEPT achieves the highest average final performance among the evaluated methods under matched target-domain interaction budgets, improving average success by 25 percentage points over the strongest baseline.

1 INTRODUCTION

Policies trained with reinforcement learning (RL) can suffer substantial performance degradation when deployed under dynamics that differ from those encountered during training (Eysenbach et al., 2021; Liu et al., 2022). Even when the task remains unchanged, differences in contact dynamics, friction, or actuator response can cause a previously successful policy to fail, particularly in contact-rich manipulation. Sim-to-real transfer is one important instance of this problem, where discrepancies between simulated and physical dynamics hinder deployment (Tobin et al., 2017; Peng et al., 2018; James et al., 2019; Akkaya et al., 2019; Andrychowicz et al., 2020). We study cross-domain policy adaptation under such dynamics shifts: given abundant interaction in a source domain but only a limited interaction budget in a target domain, how can we best exploit source-domain knowledge to learn an effective target-domain policy?

Existing approaches to addressing the dynamics gap, such as domain randomization (Tobin et al., 2017; Peng et al., 2018) and system identification (Ramos et al., 2019; Memmel et al., 2024), rely on a specified parametric family of dynamics. When this family does not adequately capture the target domain, a mismatch remains that parameter tuning alone can not solve. Model-based adaptation addresses this limitation by fine-tuning a source-pretrained world model using limited target domain data, emerging as a promising paradigm for cross-domain policy adaptation (Levy et al., 2026).

Recent work has attributed the sample efficiency of model-based methods in part to their ability to adapt the world model using limited target domain data (Wang & Johnson, 2021; Levy et al., 2026; Lanier et al., 2026; Yang et al., 2026). However, these comparisons often differ in another important aspect. In latent model-based methods, the dynamics model and value function operate over a shared learned representation, allowing the source-trained value function to be retained while the dynamics are adapted to the target domain. Model-free baselines, in contrast, are often trained with critics from scratch. It is unclear how much of the observed performance gap is attributable to the world model and how much comes from the value transfer in model-based methods.

In our work, we ask this question using a controlled cross-domain benchmark that isolates value transfer from world-model adaptation. We find that a substantial portion of the apparent sample-

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.

efficiency advantage of model-based adaptation comes from reusing the source value function, which model-free methods can also exploit. Initializing model-free baselines with the source critic allows them to adapt faster than their Dyna-style counterparts (Sutton, 1991; Feinberg et al., 2018), suggesting that world models alone do not necessarily provide higher sample efficiency.

We next evaluate when a world model provides additional benefits after controlling for value transfer. Our Dyna-style experiments highlight two complementary factors: dynamics accuracy and prior-guided data collection. Combining a dynamics ensemble with a preference for the source policy allows Dyna-style adaptation to outperform its model-free counterpart on average. Model-based planning offers another way to use the world model while retaining prior behavior through a proposal policy (Hansen et al., 2022; 2024). It has a strong initial adaptation rate, but adaptation slows or even plateaus in later stages. To identify what limits planner performance, we examine the value function, dynamics model, and proposal policy separately. We find that dynamics error is the main limiting factor, while the value-update methods we test consistently reduce performance. Fine-tuning the proposal policy on target demonstrations improves adaptation, with further gains from regularizing it toward the source policy.

Based on these findings, we build DOMAINADEPT: **Domain** **A**daptation with **D**ynamics **E**nsembles and **P**olicy **T**ethering. Following SimDist (Levy et al., 2026), we transfer a source-pretrained world model while freezing the value head during adaptation. DOMAINADEPT introduces two targeted modifications: (1) a dynamics ensemble to reduce predictive error, and (2) "tethered" policy fine-tuning, which incorporates target-domain information into the proposal policy while encouraging the planner to search regions supported by the training data of the dynamics and value functions. Under a unified interaction budget, DOMAINADEPT adapts fastest and achieves the highest success rates across all tasks among all model-free and model-based baselines, reaching an average 70% success rate with a budget of 50k transitions.

To summarize, our contributions to the problem setting of learned control in the presence of dynamics gaps are as follows. (1) We disentangle value transfer from the world model and show that transferring the source critic substantially improves model-free adaptation, allowing it to outperform the unmodified Dyna-style method, highlighting value transfer as an important confound in model-based comparisons. (2) We show that dynamics ensembles and action sampling biased towards an adapted prior policy, i.e., obtained by fine-tuning the source policy on target-domain demonstrations, provide complementary benefits in Dyna-style adaptation. (3) We propose DOMAINADEPT, which adds a dynamics ensemble and tethered proposal policy fine-tuning to model-based planning while leaving the value head frozen. Under the same target domain interaction budget, DOMAINADEPT improves success by 25% percentage points over SimDist (Levy et al., 2026).

2 RELATED WORK

2.1 OFF-DYNAMICS AND CROSS-DOMAIN POLICY ADAPTATION

Cross-domain policy adaptation addresses transition mismatches between source and target domains while keeping the task unchanged. Domain randomization perturbs source-domain dynamics during training to produce robust policies (Tobin et al., 2017; Peng et al., 2018; Tiboni et al., 2023; 2024). Its effectiveness depends on the choice of randomization range: a broad range may include implausible configurations, while a narrow one may fail to cover the target dynamics. Another approach uses target-domain data to correct for dynamics mismatch. Eysenbach et al. (2021) use domain classifiers to modify rewards and penalize transitions that are inconsistent across domains. Liu et al. (2022) and Niu et al. (2022) apply related ideas to offline and hybrid offline-online RL. Other methods filter or reweight source-domain samples using value functions (Xu et al., 2023) or learned representations (Lyu et al., 2024). Model-based adaptation instead updates a source- pretrained world model using limited target-domain data (Yin et al., 2025; Levy et al., 2026). Our work investigates what the world model contributes after controlling for value transfer and identifies the conditions under which it improves adaptation.

2.2 MODEL-BASED REINFORCEMENT LEARNING

Model-based RL learns dynamics in a latent space and uses them for decision-making (Atkeson & Santamaria, 1997; Oh et al., 2017; Ha & Schmidhuber, 2018; Hafner et al., 2019; Zhang et al., 2019;

Hafner et al., 2021; 2025; Hansen et al., 2022; 2024; Fujimoto et al., 2025). They can be categorized into Dyna-style methods and model-based planning. Dyna-style methods generate imagined rollouts to augment the data for policy and value learning (Sutton, 1991; Janner et al., 2019; Feinberg et al., 2018; Buckman et al., 2018). Model-based planning methods instead query the model at test time, using model predictive control (MPC) to search for the best action at each step (Williams et al., 2015; Nagabandi et al., 2018; Ebert et al., 2018; Chua et al., 2018; Lowrey et al., 2019; Wang & Ba, 2020; Chang et al., 2026). These two approaches have not been systematically compared under dynamics shifts, particularly against model-free RL when the target interaction budget is limited.

3 Problem Formulation

We model each domain as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $p(s'|s,a)$, reward function r, discount $\gamma \in [0,1)$ and initial state distribution ρ_0 (Sutton & Barto, 2018). Among the components of this tuple, the transition kernel (environmental dynamics) p encodes physics into the problem, including inertia, friction, contact stiffness and actuator response. The remaining components specify the task (r, γ, ρ_0) or the agent's interface to it ($\mathcal{S}$, $\mathcal{A}$). A policy $\pi(a|s)$ interacting with p induces the discounted return $J_p(\pi) = \mathbb{E}_{\pi,p}[\sum_{t\geq 0}\gamma^t r_t]$, the value functions $V_p^\pi(s) = \mathbb{E}_{\pi,p}[\sum_{t\geq 0}\gamma^t r_t \,|\, s_0 = s], Q_p^\pi(s,a) = \mathbb{E}_{\pi,p}[\sum_{t\geq 0}\gamma^t r_t \,|\, s_0 = s, a_0 = a]$. $\left(\mathcal{T}_p^\pi\, Q\right)(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}[\, Q(s', \pi(s'))]$ is the Bellman operator under p.

Our problem setting is to hold the task fixed and perturb only p. Following Kearns & Singh (2002); Lobel & Parr (2024), we instantiate two MDPs that share $(\mathcal{S}, \mathcal{A}, r, \gamma, \rho_0)$ and differ in their dynamics p_s, p_t. In a source domain with p_s, the interactions are cheap and unlimited, while a target domain with p_t is sparse. Suppose we have access to the optimal policy π_s^* and optimal value function under the environmental dynamics p_s. Since π_s was optimized against $\mathcal{T}_{p_s}$ and the contact physics it presumes, executing it under p_t generally incurs a performance drop: $\mathrm{gap}(\pi_s) = J_{p_s}(\pi_s) - J_{p_t}(\pi_s) > 0$, which we refer to as the dynamics gap. The goal is to obtain a near-optimal target policy π_t^* by transferring the source policy and value function, using only a limited rollout budget in the target domain.

We mainly compare three ways to leverage source domain data and consume target domain interaction data: (i) **Model-free value-based RL**: Off-policy RL methods can reuse data stored in their replay buffer. They fit the critic directly on collected transitions: $\min_Q \mathbb{E}_{\mathcal{D}}[(Q(s,a) - r - \gamma\bar{Q}(s', \pi(s')))^2]$, where $\bar{Q}$ is the target value, and improve policy π against it. (ii) **Dyna-style model-based RL**: Dyna-style methods additionally learn a dynamics model $\hat{p}_\theta(z' \mid z, a)$ in the latent space to generate data for RL training. They fit an approximate model $\hat{p}_\theta$ and use synthetic rollouts from $\hat{p}_\theta$ to supply extra transitions to the same value/policy update. (iii) **Model-based planning**: Model-based planning also learns a dynamics model $\hat{p}_\theta$ but uses it for action selection rather than for generating training data. The planner rolls out candidate action sequences, scores them using $\hat{r}_\phi$ and terminal value $\hat{V}_\omega$, and optimizes them with MPPI or CEM (Williams et al., 2015). Candidates are initialized from a proposal policy $\hat{\pi}_\eta$, and only the first optimized action is executed.

4 Empirical Analysis of Bridging the Dynamics Gap

To study adaptation under dynamics shifts, we establish a controlled benchmark across four contact-rich manipulation tasks from MimicGen (Mandlekar et al., 2023) with sparse rewards. The source and target domains share the same task and success predicate but differ in selected physical parameters. This setup supports low-cost evaluation of running hundreds of episodes for different methods. Moreover, it keeps the state and visual features the same across domains, allowing us to focus on adaptation to the dynamics gap.

To ensure a fair comparison of target domain adaptation efficiency, all methods can use source domain interaction to obtain their initial policies, while the target domain interaction budget is held fixed across methods. Each method is also provided with 20 demonstrations in the target domain collected by a scripted expert. We evaluate each method over at least 100 episodes and report the resulting success rate.

Table 1: **Value transfer strengthens model-free adaptation, while Dyna-style adaptation requires both a preference margin and a dynamics ensemble to surpass it. Model-based planning adapts fastest initially but shows limited further improvement.** Rows indicate the included components: Q_{source} = transferred source critic; margin = value threshold the IBRL actor must exceed; dyn. ens. = dynamics ensemble. "–" denotes inapplicable components. SR is success rate at the matched 50k target-transition budget. N_n is the target steps to first reach $n\%$ success, measured from runs continued to 300k; '$>$ 300k" indicates the threshold is never reached. TPA reports N_{20} since no run reaches 50%. Cumulative success $\Sigma\,\text{SR}_{50k} = \sum_{t\in 0,10k,\ldots,50k} \text{SR}(t)$ sums success over six evaluations. All runs start from a source policy and use the same 20 target demonstrations.

ID	Component			Threading		Coffee		Square		TPA		$\Sigma\,\text{SR}_{50k}\uparrow$
	Q_{source}	Margin	Dyn Ens.	$N_{50}/N_{80}\downarrow$	SR↑	$N_{50}/N_{80}\downarrow$	SR↑	$N_{50}/N_{80}\downarrow$	SR↑	$N_{20}\downarrow$	SR↑	(mean over tasks)
Model-free (IBRL)												
MF-1	✗	✗	–	175k/260k	0.00	205k/>300k	0.00	260k/>300k	0.01	>300k	0.00	0.00
MF-2	✓	✗	–	55k/90k	0.48	90k/200k	0.30	85k/190k	0.29	>300k	0.02	0.83
MF-3	✓	✓	–	35k/80k	0.67	65k/140k	0.44	45k/155k	0.51	**250k**	0.08	1.50
Model-based: Dyna-style (Dyna-IBRL)												
DY-1	✓	✗	✗	165k/>300k	0.11	>300k	0.04	275k/>300k	0.05	>300k	0.00	0.24
DY-2	✓	✗	✓	65k/120k	0.28	200k/>300k	0.04	105k/245k	0.13	>300k	0.00	0.39
DY-3	✓	✓	✗	40k/120k	0.58	70k/190k	0.30	50k/160k	0.52	>300k	0.01	1.43
DY-4	✓	✓	✓	30k/**70k**	0.71	50k/**135k**	0.51	40k/**85k**	0.60	>300k	0.01	1.80
Model-based: planning (SimDist)												
PL-1	✓	–	✗	**5k**/90k	**0.76**	**15k**/>300k	**0.56**	**10k**/>300k	**0.67**	>300k	**0.15**	**2.86**

4.1 Model-Free vs. Model-Based Adaptation: Do World Models Necessarily Lead to Improvements?

We compare three approaches introduced in Sec. 3. We first train an RL agent in the source domain to obtain a source expert policy π_s^e and value function Q_s^e. For the model-free method, we use IBRL (Hu et al., 2024), which fine-tunes the source expert on target demonstrations $\mathcal{D}_t$ as a prior μ and trains a critic Q_ψ from scratch. The critic is used to perform action selection between the learned actor π_ϕ and the prior at each step. To make better use of the prior, we generalize this selection with a "prior-preference margin" $m \geq 0$: the actor's action is chosen only when the critic scores it higher than the prior's by more than m,

$$a_\psi^m(s) = \begin{cases} \pi_\phi(s), & Q_\psi\big(s, \pi_\phi(s)\big) > Q_\psi\big(s, \mu(s)\big) + m, \\ \mu(s), & \text{otherwise}, \end{cases} \tag{1}$$

and original IBRL is recovered at $m = 0$. This action selection occurs in two places, and the margin can be applied independently in each:

$$\underbrace{a_t = a_\psi^m(s_t)}_{\text{Rollouts}}, \qquad \underbrace{y = r + \gamma\, Q_{\bar{\psi}}\big(s',\, a_{\bar{\psi}}^m(s')\big)}_{\text{TD target}}. \tag{2}$$

For the model-based approaches, we collect a diverse source-domain dataset $\mathcal{D}s^d$ using the source expert π_s^e together with a set of sub-optimal policies $\{\pi_s^i\}_{i=1}^N$. This provides broader state-action coverage for world model training and allows both Dyna-style adaptation and model-based planning to start from the same pretrained world model. We use $\Theta = \{E_\xi, \hat{p}_\theta, \hat{r}_\phi, \hat{v}_\omega, \hat{q}_\psi, \pi_\eta\}$ to denote all trainable parameters of the encoder, dynamics, reward, value, Q-value, and policy heads. The world model is trained with value targets relabeled by the source expert Q_s^e:

$$\begin{aligned} \mathcal{L}(\Theta) = \sum_{i=0}^{H-1} \delta^i \Big[& \underbrace{\|\hat{z}_{t+i+1} - \text{sg}(E_\xi(s_{t+i+1}))\|_2^2}_{\text{Dynamics loss}} + \underbrace{c_1\, \ell(\hat{r}_{t+i},\, r_{t+i})}_{\text{Reward loss}} \\ & + \underbrace{c_2\, \ell\big(\hat{v}_{t+i+1},\, Q_s^e(s_{t+i+1}, \pi_s^e(s_{t+i+1}))\big)}_{\text{Value loss}} + \underbrace{c_3\, \ell\big(\hat{q}_{t+i},\, Q_s^e(s_{t+i}, a_{t+i})\big)}_{\text{Q-value loss}} \\ & + \underbrace{c_4\, \mathbb{I}_e(a_{t+i})\, \|\pi_\eta(\text{sg}(z_{t+i})) - a_{t+i}\|_2^2}_{\text{Policy (or Proposal Policy) loss}} \Big], \end{aligned} \tag{3}$$

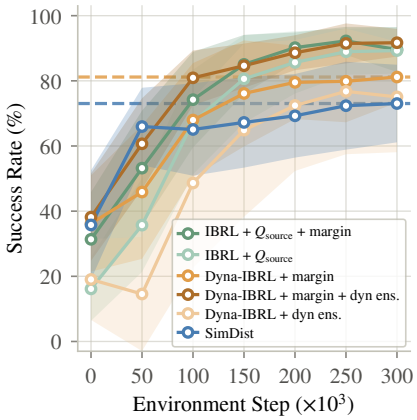

Figure 1: **Adding action sampling biased towards the prior and a dynamics ensemble allows Dyna-style adaptation to realize the advantages of the world model.** We report the mean success rates across Threading, Coffee and Square (10 seeds each), where the orange and blue dashed lines represent the performance ceilings reached by the original SimDist and Dyna-IBRL at 200k steps.

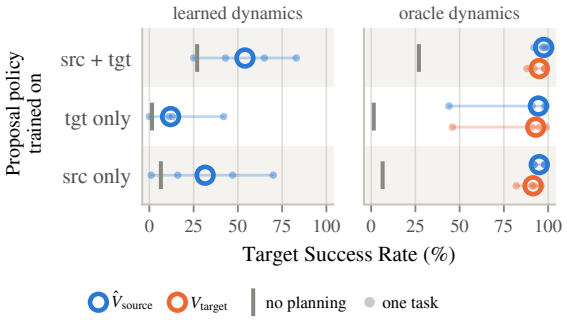

Figure 2: **Oracle dynamics enables high planning success on the target task while keeping the source value frozen.** We use the target simulation environment to replace the learned world model, called "oracle dynamics". The proposal policy is trained on "src only": the expert data from the source diverse dataset $\mathcal{D}_s^d$; "trt only": the 20 demonstrations collected from the target domain; "src + trt": the "src only" policy fine-tuned on 20 target demonstrations. $\hat{V}_{\text{source}}$ is the value trained with the world model, sharing latent space with the world model, and V_{target} is trained with the target RL expert. The points in the figure illustrate the performance of planning under $\hat{V}_{\text{source}}$ or V_{target} guidance.

where H is the rollout horizon, $\mathbb{I}_e(\cdot)$ indicates whether the action comes from the expert, ℓ denotes cross-entropy against a two-hot target obtained by linearly interpolating between the two nearest bins of a discretized scalar support (Hansen et al., 2024), and $\delta = 0.5$ discounts later rollout steps. After training, this world model is shared by both model-based variants so that the comparison isolates how the model is used rather than which model is used. Dyna-IBRL keeps training the dynamics and action-value heads while using the frozen policy head as the IBRL prior, then trains a separate actor against the critic. Planning follows SimDist (Levy et al., 2026): it transfers the value head for online search and fine-tunes only the dynamics during adaptation.

Tab. 1 shows that model-based methods adapt faster than original IBRL under the default budget (DY-1 $>$ MF-1). Some prior work attributes this advantage to the ability of model-based methods to generate additional data through a world model, thereby improving sample efficiency (Janner et al., 2019; Lanier et al., 2026). However, this comparison also *involves value transfer*. The world model carries a trained value function from the source domain, while IBRL initializes its critic from scratch. Any observed gap could be caused by the world model, the transferred value, or both. To separate them, we adapt IBRL to instead use a source expert critic Q_s^e (denoted Q_{source} in Tab. 1). This single change raises the IBRL's model-free approach cumulative success from 0 to 0.83, surpassing the basic Dyna-IBRL variant with the transferred source critic but without a margin or dynamics ensemble (MF-2 $>$ DY-1). Nonetheless, after controlling for value transfer, the best variant of model-based Dyna-style adaptation still leads (DY-4 $>$ MF-3). In the next section, we investigate what drives this residual advantage and under what conditions it emerges.

4.2 What Makes Dyna-style method Work: Accurate Dynamics and Prior-Guided Data Collection

We first ask whether reducing dynamics error alone is sufficient to close the remaining gap between model-based and model-free adaptation. We replace the single dynamics model in Dyna-IBRL with an ensemble to improve predictive accuracy. However, even with this change, Dyna-IBRL still underperforms IBRL, especially early in adaptation (MF-2 $>$ DY-2). The variant without a margin in data and TD target in Tab. 2 provides a clue to this behavior. Early in adaptation, Dyna-IBRL relies heavily on the newly initialized RL actor for both data collection and TD backup. As a result, imagined rollouts in the world model do not directly benefit from the prior actor during this stage

Table 2: **The margin helps primarily through data collection rather than TD target construction.** The prior-preference margin requires the RL actor's estimated value to exceed the prior's by at least m. It is applied in the channels marked ✓ and zeroed elsewhere. We report "SR@50k" as success rate at budget of 50k transitions on Threading. "Win rate" is the fraction of steps at which the RL actor wins the argmax, reported separately for collection and for the backup averaged at 0-50k steps.

Margin applied in				Win rate	
Rollouts	TD target	SR@50k	$N_{50} \downarrow$	Collect	Backup
✓	✓	0.711	30k	0.156	0.113
✓	✗	0.733	35k	0.116	0.815
✗	✓	0.535	50k	0.672	0.076
✗	✗	0.262	70k	0.715	0.786

Table 3: **All tested value-update schedules reduce success relative to dynamics-only adaptation.** We report the mean target success rate at the end of the target budget for five schedules across 10 seeds: (i) only dynamics adapted for 100k steps (SimDist); (ii) joint value and dynamics adapted simultaneously for 100k steps; (iii) value $\rightarrow$ dynamics: after adapting value for 100k steps, adapt dynamics for 100k steps; (iv) dynamics $\rightarrow$ value: after adapting dynamics for 100k steps, adapt value for 100k steps; (v) marginal value: the value head nudged with a small residual instead of refit.

Schedule	Threading	Square	Coffee	TPA	mean
Dynamics only	**78.7**	**72.6**	**60.2**	**16.2**	**56.9**
Joint adapt	0.0	2.5	0.0	0.0	0.6
Value $\rightarrow$ Dynamics	0.0	0.5	0.0	0.0	0.1
Dynamics $\rightarrow$ Value	0.3	0.0	4.0	0.0	1.1
Marginal value adapt	16.3	0.0	0.0	0.0	4.1

and can instead slow down adaptation. This motivates us to introduce a preference margin m, which requires the learned actor to outperform the prior by at least m before replacing it. After sweeping m (see App. D.2), we apply the margin to both data collection and TD target construction. Tab. 2 further shows that its main benefit comes from controlling the data distribution rather than from modifying the bootstrap target: setting $m = 0$ in the TD target alone has little effect, while removing it from target-environment interaction and imagined rollouts degrades performance and increases N_{50} from 30k to 50k. These results indicate that maintaining sufficient reliance on the prior during early adaptation is important primarily because it keeps the collected and imagined data closer to the prior distribution. The comparison between "IBRL + Q_{source} + margin" and "Dyna-IBRL + margin + dyn ens." in Fig. 1 and Tab. 1 (DY-4 $>$ MF-3) shows that only after combining two components does Dyna-IBRL outperform its model-free counterpart, revealing the benefit of the world model.

The preference margin, however, is not sufficient for Dyna-IBRL to outperform IBRL. When the dynamics ensemble is removed, Dyna-IBRL still fails to outperform IBRL, as shown in Fig. 1 and Tab. 1 (MF-3 $>$ DY-3). The dashed curves in Fig. 1 further show that Dyna-IBRL eventually plateaus at 80%, indicating that dynamics error limits its continued improvement even after the early-stage data-collection issue is addressed. We observe the same behavior in SimDist, where the performance of model-based planning also saturates under an inaccurate world model. These results show that model inaccuracy limits model-based adaptation: the margin improves how the model is used, but accurate dynamics are still required for imagined rollouts to provide useful learning signals.

Taken together, these results show that the advantage of world-model-based adaptation depends on two complementary conditions. The margin keeps early data collection and bootstrapping sufficiently close to the prior, preventing a newly initialized actor from dominating adaptation before it becomes reliable. The dynamics ensemble, in turn, improves the accuracy of imagined rollouts and prevents model error from limiting later-stage performance. Neither component alone is sufficient: the margin without accurate dynamics leaves the model-error ceiling unchanged, while improved dynamics without the margin cannot fully benefit from the prior during early adaptation. Their combination is what allows Dyna-IBRL to surpass the corresponding model-free baseline.

4.3 What Factors Affect Model-Based Planning?

Sec. 4.2 identifies dynamics accuracy and prior-guided data collection as complementary factors in Dyna-style adaptation. We next ask whether these findings also govern model-based planning. Although SimDist achieves the highest cumulative success within the 50k interaction budget, it plateaus at a lower performance level than the strongest Dyna-style and model-free variants as the budget grows. We therefore examine which planner components account for SimDist's rapid early adaptation, which limits its later performance (Levy et al., 2026).

We focus on three components: the terminal value $\hat{V}_\theta$ that ranks candidate trajectories, the dynamics $\hat{p}_\theta$ that rolls them out, and the proposal policy π_θ that seeds them. We leave the reward head $\hat{r}_\theta$ unchanged because it carries no signal under sparse reward. Because these components share an encoder E_θ, we freeze the remaining modules when adapting each component to isolate its effect under the same planning procedure.

Sec. 4.1 shows that transferring the source critic Q_s^e provides a large gain for IBRL. Since source-target value gap still exist, adapting value to the target domain is a natural way to improve its performance. However, Tab. 3 shows that every schedule we tried degrades performance: value first, dynamics first, the two jointly, and a deliberately marginal update interpolates toward the failure rather than avoiding it. Across the schedules, adapting the value head substantially reduces planner success relative to dynamics-only adaptation. These findings are consistent with those from SimDist (Levy et al., 2026), suggesting that retaining the source value function is preferable to adapting it under our adaptation protocol.

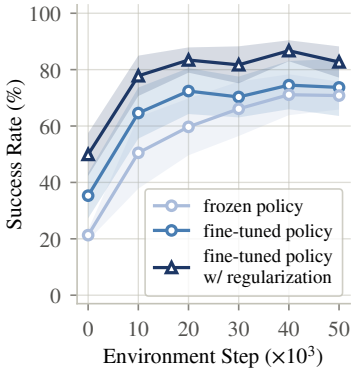

Figure 3: **Target demonstrations improve proposal initialization, and a regularization provides a further gain.** We report the mean target success rate average on Threading, Coffee and Square, with the planner's proposal policy (a) frozen at the source expert, (b) fine-tuned on the target demonstrations without regularization, and (c) fine-tuned with a regularization toward source policy.

Moreover, we conducted a simple oracle experiment in Fig. 2 to demonstrate that the unadapted source-transferred value is sufficient to achieve high planning performance with an accurate dynamics model. Replacing the learned dynamics with the target simulator substantially improves success despite not adapting the source value or the proposal policy. Under oracle dynamics, using the target value provides no clear additional benefit.

Regarding the proposal policy, Sec. 4.2 suggests that it can affect planning performance by determining the distribution of candidate actions explored by the planner. Results in Fig. 3 show that target demonstrations improve over a frozen source proposal, while a direct update is less effective than an update regularized toward the source policy. This is consistent with the benefit of prior-guided data collection observed in Sec. 4.2. While the margin controls action selection during Dyna-style adaptation, the regularization before online planning encourages retention of source behavior while incorporating target demonstrations. Therefore, the planner will search regions supported by the source training data, reducing the risk of relying on dynamics and value predictions in unfamiliar regions. Further analysis is provided in App. D.1.

5 DomainADEPT: Domain Adaptation with Dynamics Ensembles and Policy Tethering

Sec. 4.3 motivates three design choices. First, the value should be kept. Second, dynamics error determines the performance ceiling, so reducing it is a direct way to raise final performance. Third, the proposal policy must incorporate target demonstrations while staying tethered to the source, so that the dynamics and the value remain valid in the region the planner actually searches. Our method, DomainADEPT, applies two modifications to SimDist (Levy et al., 2026) by adding an ensemble for the dynamics and a "tether" on the proposal policy, and keeps freezing the value head.

Plan-time dynamics ensemble. The K dynamics heads are pretrained in the source domain with independent initialization and independently resampled training subsets obtained using Bernoulli masks, then fine-tuned on collected target transitions. At plan time, the planner rolls out on the mean prediction $\bar{f}(z,a) = \frac{1}{K}\sum_i f_i(z,a)$ rather than on any single head. Ensemble-mean planning improves performance from 82% to 96% over the single-model variant on Threading (see Sec. 6.2), consistent with the hypothesis that more accurate dynamics predictions benefit adaptation. We also tested penalizing disagreement during planning and rewarding it during exploration, but neither produces a meaningful effect (App. D.2).

Tethered proposal policy fine-tuning. An obvious way to give the proposal policy target information is to clone the target demonstrations. Due to the limited number and sparse distribution of demonstrations, directly fitting a policy on these data causes it to forget behaviors within the original dense distribution (Jain et al., 2025). Meanwhile, since the planner generates actions by iteratively adding noise to actions sampled from proposals, a policy directly fine-tuned on a small target demonstration set may alter behavior on source states that remain useful for planning. This leads to sampling states where the value lacks supervision, ultimately resulting in inaccurate value estimation. On the other hand, directly using the policy distilled from the source leads to slower adaptation without target warmup, as we discuss in Sec. 4.3. We therefore penalize deviations from the source policy on the source dataset while fitting the target demonstrations.

$$\mathcal{L}(\theta) \;=\; \underbrace{\mathbb{E}_{(s,a)\sim\mathcal{D}_t}\left[\|\pi_\theta(s)-a\|_2^2\right]}_{\text{Behavior Cloning}} \;+\; \lambda\, \underbrace{\mathbb{E}_{s\sim\mathcal{D}_s^d}[D(\pi_\theta(\cdot|s)\,\|\,\pi_s(\cdot|s))]}_{\text{Policy Tethering}}, \tag{4}$$

where λ is a hyperparameter, $\mathcal{D}_s^d$ is the source diverse dataset that the world model is trained on, and π_s is the frozen policy trained on an expert subset of $\mathcal{D}_s^d$. We call this *tethering* the proposal policy: it is free to move locally toward the target demonstrations, but is penalized for leaving the region on which the frozen dynamics and value heads were fitted. Here $D(\cdot\|\cdot)$ is a divergence between the adapted and source proposal policy distributions. We default to the squared L_2 distance $D = \|\pi_\theta(s) - \pi_s(s)\|_2^2$ and compare alternative choices in Sec. 6.2.

The value and reward heads stay frozen at their source-distilled values throughout adaptation, and the encoder is frozen after source domain training. Beyond the fixed set of target demonstrations, it requires no target reward shaping, target expert, or access to p_t at any point. All remaining architecture and optimization settings follow SimDist (Levy et al., 2026).

6 EVALUATIONS

We evaluate DOMAINADEPT through experiments addressing three questions: (1) How does DOMAINADEPT compare with existing adaptation methods under limited target interaction budgets? (2) How do dynamics ensembles and policy tethering contribute to adaptation performance? (3) How do the regularization formulation, reference actions, and state coverage affect policy tethering?

6.1 CROSS-DOMAIN ADAPTATION

Baselines. We compare DOMAINADEPT with six adaptation baselines. *Model-free*: IBRL (Hu et al., 2024) uses critic-guided selection between prior and actor actions; RLPD (Ball et al., 2023) mixes demonstrations with online data; IQL (Kostrikov et al., 2022) uses offline pretraining followed by online fine-tuning. *Model-based*: Dyna-IBRL adds model-generated transitions to IBRL; Dyna-IBRL+SGFT (Yin et al., 2025) additionally uses source-value reward relabeling; SimDist (Levy et al., 2026) is the framework underlying DOMAINADEPT and our primary model-based baseline. We also report the domain-randomized source policy's zero-shot performance. See App. C.4 for implementation details.

We compare the methods on five tasks under a target-domain interaction budget of 50k steps per task, except for TPA, where we use 100k steps because its episodes are longer. Fig. 4 reports performance throughout adaptation. DOMAINADEPT adapt fastest and continues to improve within the allotted budget, reaching the highest average final success rate of approximately 70%.

6.2 COMPONENT ABLATION

To answer the Question (2), we ablate the two core components, i.e., the dynamics ensemble and policy tethering, of DOMAINADEPT by removing each in turn (Fig. 5). Tethering provides a substantial improvement at initialization, while the ensemble further improves performance both before and during online adaptation. The full method achieves the highest performance over the evaluated interaction budget, highlighting the complementary benefits of the two components.

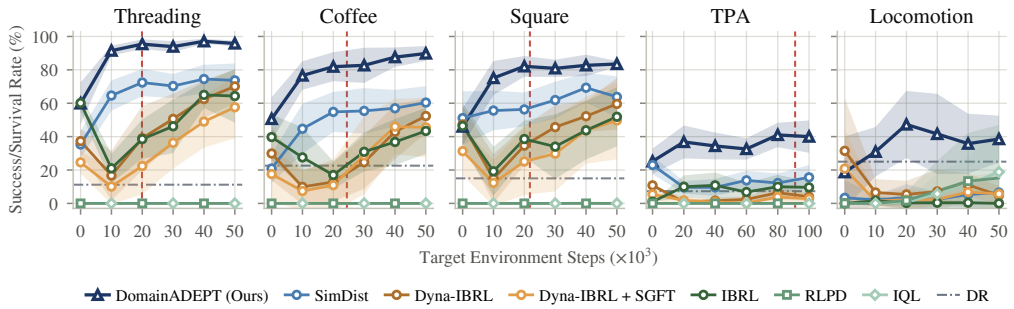

Figure 4: **DOMAINADEPT achieves the highest adaptation performance across the five evaluated tasks.** The grey dash-dotted line is the domain-randomized source baseline. Shading is ± 1 standard deviation across 10 seeds, and each point is an independent re-evaluation over 100 episodes. The red dashed line marks approximately 100 episodes of interaction for each manipulation task.

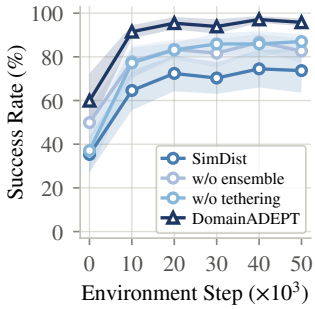

Figure 5: **Combining dynamics ensembles and policy tethering gives the best performance.** Recall that DOMAINADEPT = SimDist with dynamics ensemble and policy tethering.

Table 4: **Variants of tethering formulations on Threading.** We compare regularization formulations, state coverage, and reference actions after 50k target-domain transitions. The default regularizes outputs toward the source policy on the full source dataset; variants use successful expert states or recorded actions. Coefficients match source-policy deviation D_{src} where possible. Results show mean $\pm$ one standard deviation across seeds, with 100 evaluation episodes.

Tether	Coefficient	D_{src}	SR (%) @ 50k
(i) Regularization formulations			
Squared L_2 / KL (default)	λ=1	0.025	**95.8** $\pm$1.1
Saturating (Hellinger^2)	λ=0.013 τ=0.03	0.025	**96.0** $\pm$1.4
Trust region, tight	λ=0.3 ϵ=0.05	0.027	93.6 $\pm$3.1
Trust region, wide	λ=100 ϵ=0.20	0.031	82.9 $\pm$5.9
Parameter-space L_2	λ=0.25	0.025	72.1 $\pm$6.1
Weight decay	wd=10^{-3}	0.209	77.7 $\pm$8.7
(ii) State coverage			
Expert $\wedge$ Success	λ=1	0.026	93.7$\pm$3.7
(iii) Reference actions			
Recorded dataset action	λ=3	0.154	74.5 $\pm$6.3
Recorded, Expert $\wedge$ Success	λ=30	0.029	78.7 $\pm$6.0

6.3 POLICY TETHERING DESIGN CHOICE

To answer the Question (3), we compare different tethering designs in Tab. 4. We tune their coefficients to match the source-policy deviation D_{src} where possible (App. C.6). Squared L_2, the saturating penalty, and the tight hinge achieve similar success rates (95.8%, 96.0%, and 93.6%). A wider hinge performs worse (82.9%), as does parameter-space L_2 (72.1%). These results demonstrate that regularizing policy outputs performs better than regularizing policy parameters. The choice of reference actions also matters. Using source-policy outputs on only expert-success states still achieves 93.7%, close to the default 95.8%. Replacing these outputs with recorded actions reduces success to 78.7% on the same subset and 74.5% on the full dataset. These results support using the source policy's output as the target. Broader coverage in the tethering dataset provides an additional performance gain and is required to achieve the best performance.

7 CONCLUSION

In our paper, we show that value transfer is an important factor in comparing model-free and model-based adaptation. Our experiments highlight the complementary benefits of dynamics accuracy and prior-guided data collection, motivating DOMAINADEPT: dynamics ensembles and tethered proposal fine-tuning with a frozen source value. DOMAINADEPT improves adaptation under limited target interaction budgets, demonstrating the benefit of selectively adapting transferred components.

Ethics Statement

This work adheres to the ICLR Code of Ethics. No human participants or animals were involved in this study. All datasets used in our experiments were obtained and used in accordance with their relevant licenses and usage guidelines, with no violation of privacy. No personally identifiable information was used, and no experiments were conducted that raise privacy or security concerns. We also took care to avoid introducing unfair bias or discriminatory outcomes throughout the research process. We are committed to maintaining transparency and research integrity throughout this work.

LLM Usage

Large Language Models (LLMs) were used to aid in the writing and polishing of the manuscript. Specifically, we used an LLM to assist in refining the language, improving readability, and ensuring clarity in various sections of the paper. The model helped with tasks such as sentence rephrasing.grammar checking, and enhancing the overall flow of the text.

It is important to note that the LLM was not involved in the ideation, research methodology, or experimental design. All research concepts, ideas, and analyses were developed and conducted by the authors. The contributions of the LLM were solely focused on improving the linguistic quality of the paper, with no involvement in the scientific content or data analysis.

The authors take full responsibility for the content of the manuscript, including any text generated polished by the LLM. We have ensured that the LLM-generated text adheres to ethical guidelines and does not contribute to plagiarism or scientific misconduct.

References

Pieter Abbeel, Morgan Quigley, and Andrew Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pp. 1–8, 2006.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Eugene Ribakov, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Przemyslaw Dkebiak. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. doi: 10.1177/0278364919887447.

Christopher G. Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pp. 3557–3564. IEEE, 1997.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1577–1594. PMLR, 2023.

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 (NeurIPS)*, volume 31, 2018.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*. PMLR, 2026.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979. IEEE, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *International Conference on Learning Representations (ICLR)*, 2021.

Amir-massoud Farahmand. Iterative value-aware model learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, 2017.

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.

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

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

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 8387–8406, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. In *Proceedings of Robotics: Science and Systems (RSS)*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.056.

Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled sailor: Robust imitation via learning to search. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 62676–62707, 2025.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12619–12629. IEEE, 2019.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. TRANSIC: Sim-to-real policy transfer by learning from online correction. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1691–1729. PMLR, 2025. URL https://proceedings.mlr.press/v270/jiang25a.html.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2):209–232, 2002.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

JB Lanier, Kyungmin Kim, Armin Karamzade, Yifei Liu, Ankita Sinha, Kat He, Davide Corsi, and Roy Fox. Adapting world models with latent-state dynamics residuals. In *Proceedings of the 8th Annual Learning for Dynamics and Control Conference*, pp. 117–144, 2026.

Jacob Levy, Tyler Westenbroek, Kevin Huang, Fernando Palafox, Patrick Yin, Shayegan Omidshafiei, Dong-Ki Kim, Abhishek Gupta, and David Fridovich-Keil. Simulation distillation: Pre-training world models in simulation for rapid real-world adaptation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2026.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Jinxin Liu, Hongyin Zhang, and Donglin Wang. DARA: Dynamics-aware reward augmentation in offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Sam Lobel and Ronald Parr. An optimal tightness bound for the simulation lemma. *Reinforcement Learning Journal*, 2:785–797, 2024.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations (ICLR)*, 2019.

Jiafei Lyu, Chenjia Bai, Jing-Wen Yang, Zongqing Lu, and Xiu Li. Cross-domain policy adaptation by capturing representation mismatch. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33638–33663. PMLR, 21–27 Jul 2024.

Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, Los Angeles, CA, USA, 2025. doi: 10.15607/RSS.2025.XXI.109.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Dieter Fox, and Abhishek Gupta. ASID: Active exploration for system identification in robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2024.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

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. Citeseer, 1999.

Haoyi Niu, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, Xianyuan Zhan, et al. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36599–36612, 2022.

NVIDIA. Isaac Sim. URL https://github.com/isaac-sim/IsaacSim.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pp. 5062–5071. PMLR, 2019.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *Proceedings of Robotics: Science and Systems (RSS)*, Freiburg im Breisgau, Germany, 2019. doi: 10.15607/RSS.2019.XV.029.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pp. 8583–8592. PMLR, 2020.

Liangzhi Shi, Shuaihang Chen, Feng Gao, Yinuo Chen, Kang Chen, Tonghe Zhang, Hongzhi Zang, Weinan Zhang, Chao Yu, and Yu Wang. Beyond imitation: Reinforcement learning-based sim-real co-training for VLA models. *arXiv preprint arXiv:2602.12628*, 2026.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International conference on machine learning*, pp. 5779–5788. PMLR, 2019.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

Gabriele Tiboni, Karol Arndt, and Ville Kyrki. Dropo: Sim-to-real transfer with offline domain randomization. *Robotics and Autonomous Systems*, 166:104432, 2023. doi: 10.1016/j.robot.2023.104432.

Gabriele Tiboni, Pascal Klink, Jan Peters, Tatiana Tommasi, Carlo D'Eramo, and Georgia Chalvatzaki. Domain randomization via entropy maximization. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=GXtmuiVrOM.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE, 2017.

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.015.

Claas A. Voelcker, Anastasiia Pedan, Arash Ahmadian, Romina Abachi, Igor Gilitschenski, and Amir-Massoud Farahmand. Calibrated value-aware model learning with probabilistic environment models. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 61745–61768, 2025.

Sean J Wang and Aaron M Johnson. Domain adaptation using system invariant dynamics models. In *Learning for Dynamics and Control*, pp. 1130–1141. PMLR, 2021.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Kang Xu, Chenjia Bai, Xiaoteng Ma, Dong Wang, Bin Zhao, Zhen Wang, Xuelong Li, and Wei Li. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 36:73395–73421, 2023.

Wentao Xu, Huiqiao Fu, Haoyu Dong, Zhehao Zhou, and Chunlin Chen. Deal: Diffusion evolution adversarial learning for sim-to-real transfer. In *Advances in Neural Information Processing Systems*, volume 38, pp. 146698–146724, 2026.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, Ping Luo, Xiangyu Yue, and Hongyang Li. RISE: Self-improving robot policy with compositional world model. In *Proceedings of Robotics: Science and Systems (RSS)*, 2026.

Patrick Yin, Tyler Westenbroek, Ching-An Cheng, Andrey Kolobov, and Abhishek Gupta. Rapidly adapting policies to the real world via simulation-guided fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2025.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. SOLAR: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7444–7453. PMLR, 2019.

A Extended Related Work

A.1 Additional approaches to cross-domain adaptation

System identification fits a parameterized dynamics model using target-domain observations (Ramos et al., 2019; Chebotar et al., 2019; Memmel et al., 2024; Xu et al., 2026). Its effectiveness depends on the chosen parameterization: effects such as contact transients and nonlinear friction may remain difficult to capture, leaving residual errors that affect policy transfer. Other approaches adapt policies through target domain interaction (Torne et al., 2024), jointly train on source and target domain data (Maddukuri et al., 2025; Shi et al., 2026), or use human corrections during execution (Jiang et al., 2025). These approaches differ in how they obtain and use target domain information. Our study focuses on adaptation with a fixed set of target demonstrations and a limited online interaction budget, investigating the contribution of world models after controlling for value transfer.

A.2 Model error, ensembles and disagreement

Many prior works have studied how to address inaccuracies in learned dynamics models use bootstrap resampling in model ensembles to decorrelate individual models (Abbeel et al., 2006). Lakshminarayanan et al. (2017), while using the ensemble mean as a low-variance predictor to improve training stability (Kurutach et al., 2018). This property of ensemble models is a key component on which DomainADEPT relies. In addition, some methods use the uncertainty of ensemble models as a penalty to constrain learning to regions where the model is reliable (Kidambi et al., 2020; Yu et al., 2020; 2021). Other methods instead use such uncertainty as an intrinsic reward to guide exploration, encouraging more learning in regions with larger model errors to reduce their impact (Pathak et al., 2019; Shyam et al., 2019; Sekar et al., 2020).

We experimented with both strategies but did not observe significant improvements (see Sec. D.2). One plausible explanation is that the tether mechanism already restricts the planner to regions covered by the source dynamics model. As a result, there is little room for pessimistic penalties to take effect, while exploration bonuses also have limited contribution.

B Environmental details



(a) Threading (b) Square (c) Coffee (d) TPA (e) Locomotion

Figure 6: **The five evaluation environments.** Four contact-rich manipulation tasks from MimicGen and one locomotion task from IsaacLab.

We use four contact-rich manipulation tasks from MimicGen (Mandlekar et al., 2023), including Threading, Square, Coffee and ThreePieceAssembly, and one locomotion task from IsaacLab (NVIDIA). The four manipulation tasks share one shift mechanism (multiplicative changes to physical parameters), and the locomotion task instead changes terrain geometry. All five are simulated, which is what lets us run several hundred evaluation episodes per condition and, more importantly, run the same controller with the ground-truth target dynamics in place of the learned model (the oracle-dynamics in Sec. 4.3).

Each task gives us a source domain and a target domain. The source domain is the nominal simulator as shipped. The target domain is the same simulator with a fixed perturbation applied to its physical parameters: we scale object mass and inertia, the sliding friction of the manipulated object and of the fixture it must be inserted into, the stiffness of the gripper actuator, and the geometric clearance of the insertion. The task, the assets, the observation space and the success predicate are untouched. This is a deliberate choice: because the two domains differ only in physics, a policy that fails after

transfer fails for a dynamics reason and not because it is looking at a different scene or being scored by a different rule.

Two consequences of that choice are worth stating explicitly. First, the success predicate is a fixed geometric tolerance on the object pose and is not re-derived from the perturbed model, so shrinking an insertion clearance makes the task physically harder without redefining the success condition. Second, the object's initial pose is drawn from the same distribution in both domains, so the gap is not a perception or registration problem.

Table 5 lists the perturbation for each manipulation task. Every entry is a multiplicative scale on the corresponding field of the MuJoCo model, so 1.0 means unchanged. The scales were chosen per task to produce a drop such that the frozen source expert loses most of its success rate in the target domain, while a policy trained directly in the target domain can still solve the task, which is what makes the remaining gap attributable to adaptation rather than to task difficulty.

Table 5: Target-domain physical perturbations. Values are multiplicative scales on the nominal simulator (1.0 = unchanged; — = not applicable to that task). "fixture friction" and "clearance" refer to the part the object is inserted into the ring for Threading, the peg for Square, the pod holder for Coffee, and the base hole for ThreePieceAssembly. σ is the standard deviation of a Gaussian perturbation added to the robot's initial joint configuration, applied identically in both domains. †For Square the scale is applied to the peg rather than to the hole, so a value *above* one shrinks the clearance; the effect is the same as the sub-one scales elsewhere in that row.

	Threading	Square	Coffee	ThreePiece
object mass & inertia	3.0	1.0	8.0	1.0
object friction	0.5	1.0	0.2	1.0
fixture friction	2.0	2.0	5.0	2.0
insertion clearance	0.85	$1.2^{\dagger}$	1.0	0.96
gripper stiffness	0.12	0.3	0.3	0.25
lid hinge damping	—	—	5.0	—
episode length	400	400	400	1000
initial-pose noise σ	0.1	0.1	0.1	0

The locomotion environment is a Unitree Go2 quadruped walking under a velocity command in IsaacLab (NVIDIA). The policy observes a 35-dimensional proprioceptive vector (base linear and angular velocity, projected gravity, joint positions and velocities, the previous action, and a cosine/sine gait phase) together with a 315-point height scan of the terrain under the robot, and outputs 12 joint position targets at 50Hz. Episodes last 1000 control steps and terminate early on a base collision or on the trunk tipping past its orientation limit.Unlike manipulation's sparse reward, Go2's locomotion task utilizes a dense reward function that continuously evaluates tracking performance, gait shaping, smooth control, and survival into a single cumulative score per episode.

The shift is geometric rather than parametric. The source domain uses flat ground, while the target domain replaces it with a field of boxes at difficulty 0.7, where box height and spacing increase with difficulty. The robot, command distribution, observation space, reward, and contact parameters remain unchanged. What changes is the contact pattern that the gait must handle. On flat ground, foot touchdown is largely predictable from the joint state, whereas on the boxes, touchdown height varies across feet and steps and is available to the policy only through the height scan. This provides a locomotion analogue of the manipulation shifts: the task and reward remain unchanged, while performance is affected by a change in the underlying dynamics.

Target-domain demonstrations come from a PPO expert trained under domain randomization with privileged observations (terrain friction, added base mass and actuator properties, none of which the distilled model sees). The expert still walks on the target terrain where the distilled policy falls, which is what makes it usable as a demonstrator. Each seed draws 20 expert episodes from 20 distinct box layouts; evaluation uses five further layouts, disjoint from both the demonstration layouts and the layout the online phase trains on, so nothing is scored on a layout it was fit to.

Table 6: Locomotion source and target domains. The shift is terrain geometry. Every contact and actuator parameter is identical in the two domains. Terrain difficulty interpolates box height and spacing. Layout seeds index the procedural terrain generator: demonstrations, online training and evaluation use disjoint sets, so no reported number comes from a layout that was fit to.

	Source	Target
terrain	flat	boxes
terrain difficulty	0.4	0.7
foot friction	0.8	0.8
restitution	0.05	0.05
added base mass (kg)	0	0
observation (proprio + height scan)	$35 + 315$	
action (joint position targets)	12	
command (forward velocity, m/s)	0.5	
episode length (steps)	1000	
metric	survival rate	

C METHOD DETAILS

C.1 ALGORITHM

DOMAINADEPT runs in three stages. Stage 1 happens entirely in the source domain and is paid for once. Stages 2 and 3 are the adaptation proper and are the only parts that consume the target budget.

Stage 1: source domain world model distillation. We first train an IBRL agent to convergence in the source domain, which gives us an expert policy π_s^e and its critic Q_s^e. We then roll out that expert together with a set of weaker checkpoints $\{\pi_s^i\}_{i=1}^N$ saved along its training curve, and collect a large replay of source trajectories. We then distill this replay into a latent world model with five heads: an encoder, a dynamics head, a reward head, a state value head and a state-action head. In parallel, a policy head is trained separately on expert demonstrations via behavior cloning. All of them are trained jointly with the multi-step loss of Eq. 3. The value targets are relabeled by the frozen source critic, so the value head learns $V^{\pi_s^e}$ rather than a return estimate from sparse rewards.

Two details are important. The dynamics head is an ensemble of K heads that share the encoder but are independently initialized and trained on independent bootstrap subsets of each batch, so they disagree off the data manifold. And the gradient of the policy-cloning term is stopped before it reaches the encoder, so the policy cannot reshape the latent space that the dynamics depends on.

Stage 2: warm up the proposal policy on target demonstrations, tethered to the source. We use the 20 target demonstrations and no target interaction. We fine-tune only the policy head, minimizing the behavior cloning loss plus the tether penalty of Eq. 4. The encoder, the dynamics, the reward and the value heads are all frozen.

Stage 3: target domain dynamics adaptation. We alternate collection and dynamics fitting in rounds. In each round the agent acts in the target domain with the MPPI planner for a fixed number of transitions, appends them to a replay buffer, and then takes a number of gradient steps on the dynamics heads only, using the same latent-consistency loss as stage 1, restricted to the dynamics term. The encoder, the reward head, the value head and the policy head all stay frozen for the entire stage.

C.2 MODEL STRUCTURE

For tasks with state-based observations, the encoder maps the raw state to a 512-dimensional latent normalized by SimNorm (Hansen et al., 2024) (softmax over groups of 8), which prevents magnitude drift during multi-step rollouts. The dynamics module is an ensemble of K MLP heads over this shared encoder, each predicting the next SimNorm'd latent; planning rolls out on their mean. The reward and value heads predict distributions over 101 bins via two-hot cross-entropy rather than scalars, and the planner uses the expected value. The policy head maps the latent to an action with

Algorithm 1 DoaminADEPT

Require: source domain $\mathcal{M}_s$, target domain $\mathcal{M}_t$, target demonstrations $\mathcal{D}_t$, budget N, dynamics ensemble number K

Stage 1: source distillation

train an RL expert (π_s^e, Q_s^e) in $\mathcal{M}_s$

collect diverse dataset $\mathcal{D}_s^d$ with π_s^e and weaker checkpoints $\{\pi_s^i\}_{i=1}^N$ of it

distill $\{E_\theta, \hat{p}_\theta^{1..K}, \hat{r}_\theta, \hat{V}_\theta, \pi_\theta\}$ on $\mathcal{D}_s$ with Eq. 3; relabel values with Q_s

Stage 2: proposal policy warm-up

fine-tune π_θ only, on $\mathcal{D}_t$ with the tethering on $\mathcal{D}_s^d$ (Eq. 4)

Stage 3: dynamics adapt

$\mathcal{R} \leftarrow \mathcal{D}_t$; $i \leftarrow 0$

while $i < N$ **do**

  collect m transitions in $\mathcal{M}_t$ with MPPI, ensemble-mean rollout; append to $\mathcal{R}$; $i \leftarrow i + m$

  take u gradient steps on $\hat{p}_\theta^{1..K}$ only, latent-consistency loss on $\mathcal{R}$ ▷ $E_\theta, \hat{r}_\theta, \hat{V}_\theta, \pi_\theta$ frozen

end while

return planner using $\{E_\theta, \hat{p}_\theta^{1..K}, \hat{r}_\theta, \hat{V}_\theta, \pi_\theta\}$

the cloning gradient stopped before the encoder. A decoder trained on detached encodings inverts the encoder for reconstruction but is not used by the planner in the default configuration.

For pixel-based observations, we replace the state encoder with a visual encoder while keeping the latent space, dynamics ensemble, and reward, value, and policy heads unchanged. Each observation consists of two 224×224 RGB views, one third-person view and one wrist-camera view, together with the 9-dimensional proprioceptive state. Object-state information is therefore inferred from vision. Each camera is processed by an ImageNet-pretrained ResNet-18, and the resulting visual features are concatenated with proprioception and projected into the same 512-dimensional SimNorm latent space used in the state-based setting. We do not use an image decoder. During stage-1 training, the cloning loss is allowed to update the encoder at the first rollout step, while later rollout steps remain stop-gradient as in the state-based setting. We apply standard image augmentations, including random shifts, color jitter, blur, and proprioceptive noise, with augmentation parameters held fixed across each training sequence to preserve temporal consistency.

C.3 Hyperparameter

Table 7 lists the hyperparameters. They are shared across all four manipulation tasks except for the planner's search parameters, which we re-tune per task, since we found them not to transfer, including between a learned and an oracle dynamics model, and we report the swept values in Table 8.

C.4 Baseline details

IBRL. IBRL is our model-free reference. Following Hu et al. (2024), a behaviour-cloning prior μ is obtained by fine-tuning the source expert on the target demonstrations and then frozen. An actor π_ϕ and a twin critic Q_ψ are trained online with TD3-style updates, and at every step the executed action is the critic's argmax over the two candidates $\{\pi_\phi(s), \mu(s)\}$ (Eq. 2). The same argmax is used inside the Bellman backup, so the prior bootstraps the value function rather than only performing as an actor. The demonstrations are kept in a frozen half of the replay buffer like RLPD (Ball et al., 2023), so every minibatch is half demonstration data and half online data throughout training.

We report three variants, which differ in exactly one thing each:

- **IBRL**: the original method, and the critic is initialized from scratch.
- **IBRL + source critic**: the critic is initialized from the source expert's critic Q_s instead.
- **IBRL + source critic + margin**: additionally requires the learned actor to beat the prior by a margin m before it is executed:

$$a(s) = \begin{cases} \pi_\phi(s) & \text{if } Q_\psi(s,\, \pi_\phi(s)) > Q_\psi(s,\, \mu(s)) + m, \\ \mu(s) & \text{otherwise.} \end{cases} \tag{5}$$

Table 7: DoaminADEPT hyperparameters.

Stage	Parameter	Value
1 (distillation)	latent dimension	512
	SimNorm group size	8
	MLP hidden size / layers	512 / 2
	rollout horizon H	5
	per-step loss discount δ	0.5
	dynamics ensemble size K	5
	reward/value bins, range	101, $[-10, 10]$
	loss weights (dyn. / rew. / val. / BC)	20 / 0.1 / 0.1 / 10
	learning rate	3×10^{-4}
	batch size	256
	gradient steps	800k
	source replay size	$\sim$55k episodes
2 (proposal policy warm-up)	target demonstrations n	20
	gradient steps	1000
	learning rate, batch size	3×10^{-4}, 256
	tether strength λ	$\{0.1, 0.3, 1.0, 3.0, 10\}$
3 (dynamics adapt)	target budget N	$50/100$k transitions
	transitions per round m	1000
	dynamics steps per round u	1000
	learning rate, batch size	3×10^{-4}, 256

Table 8: MPPI search parameters. `init_std` and the base-policy noise were re-swept per task; the remaining entries are the same everywhere.

Parameter	Value
candidate sequences per iteration	350 (100 from the proposal policy, 250 random)
planning horizon	5
CEM iterations	3
elites	64
softmax temperature	0.4
discount	0.99
initial std / minimum std	$\{0.15, 1.0\}$ / 0.05
base-policy noise std	$\{0.01, 0.1\}$

Dyna-IBRL. The model-based counterpart to IBRL, and it isolates how a world model is used. It shares the same distilled world model, the same frozen prior from the same warm-up with our method and differs only in what happens during stage 3. Instead of planning with the model at decision time, it uses the model as a data generator: each round it branches imagined rollouts from real latents, adds them to a synthetic buffer, and trains an actor and a $Q(z, a)$ head on a mixture of real and imagined transitions. Imagination is carried out entirely in latent space rather than decoding to states and re-encoding, because that round trip is not the identity and it will increase the error (Hafner et al., 2021). Concretely, from each real latent $z_t = E_\theta(s_t)$ in the replay buffer, the model branches H-step imagined trajectories:

$$\hat{z}_{t+k+1} = f_\theta(\hat{z}_{t+k},\, \hat{a}_{t+k}), \quad \hat{a}_{t+k} = a_\psi(\hat{z}_{t+k}), \quad k = 0, \ldots, H-1, \tag{6}$$

where a_ψ is the IBRL argmax of Eq. 5. The imagined transitions are added to a synthetic buffer, and the actor and critic are trained on a mixture of real and imagined data.

SGFT. Simulation-guided fine-tuning (Yin et al., 2025) transfers the source value function as a dense reward instead of as a critic initialization, because the source value already encodes the long-horizon structure of the task, so the target-domain run only has to correct behaviour over a short horizon. We implement the potential-based form used in the Yin et al. (2025) implementation, which makes it a simple change from Dyna-IBRL: the reward entering the critic's Bellman backup is relabeled, and the critic is reinitialized. Writing $\Phi(z) = \hat{V}_\theta(z)$ as the trained source value function,

the reward is relabeled as

$$\tilde{r}_t \;=\; r_t \;+\; \alpha\big(\gamma\,(1-d_t)\,\Phi(z_{t+1}) \;-\; \Phi(z_t)\big), \qquad \Phi(z)=\hat{V}_\theta(z), \tag{7}$$

where α is the shaping weight and d_t is the terminal flag. Because the shaping is potential-based, the optimal target policy is unchanged (Ng et al., 1999). It only redistributes the sparse terminal reward across the trajectory, which a $0/1$ signal with $\gamma = 0.99$ would otherwise propagate very slowly over an episode. The effect on the critic is equivalent to warm-starting it at $\alpha\hat{V}_\theta$ (Wiewiora, 2003), so the Bellman target clamp must be shifted from $[0, 1]$ to $[-\alpha, 1]$ to avoid truncating the shaped values.

RLPD. An off-policy method designed to learn efficiently from a mixture of offline and online data (Ball et al., 2023). Following the original recipe, we sample every minibatch half offline and half online, and keep others the same as in IBRL. The offline buffer is the union of a large set of source rollouts and the 20 target demonstrations, merged into a single prior dataset. This gives RLPD more offline data than any other baseline. The critic is updated on the mixed batch:

$$\mathcal{L}(Q)=\tfrac{1}{2}\,\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{off}}}\big[(Q(s,a)-y)^2\big]+\tfrac{1}{2}\,\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{on}}}\big[(Q(s,a)-y)^2\big], \quad y=r+\gamma\,\bar{Q}(s',\pi(s')). \tag{8}$$

IQL. An offline RL method (Kostrikov et al., 2022) is used here as a warm start, followed by online fine-tuning. We first run IQL offline on source rollouts plus the target demonstrations. The value function is fit by expectile regression to avoid querying out-of-distribution actions:

$$\mathcal{L}(V)=\mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[|\tau-\mathbf{1}(Q(s,a)-V(s)<0)|\,\big(Q(s,a)-V(s)\big)^2\Big], \tag{9}$$

where $\tau \in (0, 1)$ is the expectile. The policy is then extracted by advantage-weighted regression:

$$\mathcal{L}(\pi)=\mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[\exp\big(\beta\,(Q(s,a)-V(s))\big)\,\|\pi(s)-a\|_2^2\Big]. \tag{10}$$

The resulting actor and critic then initialize an online run with the same budget and environment as others.

C.5 Computational cost

All cross-domain adaptation experiments were run on a workstation with four RTX Pro 6000 Blackwell GPUs (96GB each). The control loop runs at 20Hz, giving a 50ms budget per step. DomainADEPT's p95 latency uses about 40% of this budget in both the state and pixel settings. It is about $14\times$ slower than an MLP policy in the state setting, but the absolute latency remains small relative to the control period. Table 10 is the value used in our analysis.

The planner is cheap in width and expensive in depth. Increasing the number of candidate sequences from 100 to 350 is almost free at batch size 1. In contrast, the number of MPPI iterations and the rollout horizon scale almost linearly. The ensemble size also scales nearly linearly, adding about 1.8ms per additional head in the current implementation, because `next_all` evaluates heads serially in Python. This is an implementation cost rather than an algorithmic requirement. We can reduce this by batching the heads or capturing the rollout in a CUDA graph.

C.6 Tether variants

We define the nine variants compared in Tab. 4 and state the calibration protocol that makes them comparable.

Every variant is an instance of

$$\mathcal{L}_{\text{tether}}(\theta) \;=\; \lambda\,\mathbb{E}_{s\sim\mathcal{D}}\Big[\,g\big(\|M\,(\pi_\theta(s)-\pi_{\text{ref}}(s))\|_2\big)\Big], \tag{11}$$

which exposes exactly four free positions: the shape g of the penalty, the state distribution $\mathcal{D}$ on which proximity is measured, the metric M on action space, and the reference π_{ref} the proposal policy is pulled towards. The deployed setting is $g(x) = x^2$, $\mathcal{D} = \mathcal{D}_s^d$, $M = I$ and $\pi_{\text{ref}} = \pi_s$. Block (i) of Tab. 4 varies g and M, block (ii) varies $\mathcal{D}$, and block (iii) varies π_{ref}.

Table 9: **Baseline hyperparameters.** The upper block is shared by every online methods. The lower blocks give what is specific to each method. $\{\}$ means chooses the best parameter for each task.

Shared by all baselines	
warm-up episodes (counted against the budget)	30
batch size	256
learning rate	3×10^{-4}
discount γ	0.99
critics (clipped double-Q)	2
critic updates per environment step	5
actor update frequency	every 2 critic updates
n-step return	3
exploration noise std	0.1
demonstration fraction per batch	0.5
target-network rate τ	0.01
IBRL	
prior-preference margin m	0 / 0.01 (two variants)
Dyna-IBRL	
imagined transitions per round	20k
imagination rollout length	$\{1, 3, 5\}$
synthetic buffer size	100k
real fraction per batch	0.5
actor updates per round	500
critic updates per round	5000
collection noise std	0.1
target-policy smoothing std / clip	0.1 / 0.3
TD-target clamp	$[0,\ 1]$
Dyna-IBRL + SGFT	
shaping weight α	$\{0.5, 1.0, 2.0\}$
TD-target clamp	$[-\alpha,\ 1]$
IQL (offline warm start, then online)	
expectile τ_{exp}	0.9
offline critic steps	80k
offline actor steps	20k
weighting temperature β	3.0

Table 10: **Single-step inference latency.** Timing includes copying one observation from CPU to GPU, computing the action, and copying the action back to CPU. Each planner configuration was run three times on each of two GPUs, for 1000 steps per run after discarding the first 50 warm-up steps. MPPI uses the paper setting: 350 candidates, 3 iterations, horizon $H = 5$, and 64 elites.

Method	median (ms)	p95 (ms)
DOMAINADEPT, state, MPPI $K = 5$	11.7	20.2
SimDist, state, MPPI $K = 1$	4.8	9.5
Policy head only, state	0.86	1.6
DOMAINADEPT, pixel ResNet-18@224, MPPI $K = 5$	13.5	22.3
SimDist, pixel ResNet-18@224, MPPI $K = 1$	6.5	11.6
Pixel policy head only	1.8	3.7

The policy head is a deterministic $\tanh$-MLP, so the only way to define a divergence between π_θ and π_s is to lift both to $\mathcal{N}(\pi(s), \sigma^2 I)$ with a constant σ. Writing $d = \|\pi_\theta(s) - \pi_s(s)\|_2$, every standard divergence between two equal-covariance Gaussians is a function of d alone:

$$\mathrm{KL} = \frac{d^2}{2\sigma^2}, \quad W_2^2 = d^2, \quad H^2 = 1 - e^{-d^2/8\sigma^2}, \quad \mathrm{TV} = 2\Phi\Big(\frac{d}{2\sigma}\Big) - 1, \quad \chi^2 = e^{d^2/\sigma^2} - 1, \tag{12}$$

Table 11: **Where MPPI time is spent.** Each block varies one parameter while holding the others at the default planner setting. Median latency is reported in milliseconds.

Sweep	median latency (ms)
Candidates: 100, 350, 1000, 3000, 10000	11.2, 11.5, 18.6, 47, 153
Iterations: 1, 3, 6, 12	5.7, 11.4, 21.8, 40.0
Horizon: 1, 3, 5, 8	3.1, 7.3, 11.4, 17.9
Dynamics heads K: 1, 2, 3, 5	4.8, 6.5, 8.3, 12.2

where Φ the standard normal CDF (Cumulative Distribution Function). For small updates ($d \ll \sigma$), all five share the identical leading-order Taylor expansion $\mathcal{O}(d^2/\sigma^2)$, so constant differences are fully absorbed by re-tuning the scalar multiplier λ. Crucially, when deviations grow comparable to or exceed σ, these divergences do not span arbitrary behaviors. Instead, their growth rates relative to d^2 partition into three qualitative regimes, and our ablation picks the canonical, numerically stable representative for each:

(i) Penalty shape $g(d)$ across curvature regimes. Let $d(s) = \|\pi_\theta(s) - \pi_s(s)\|_2$.

- **Quadratic / Elastic (KL, W_2^2, default):** $g(x) = x^2$, yielding $\lambda\,\mathbb{E}_s[d(s)^2]$. Because KL and W_2^2 differ by only a scalar constant $1/(2\sigma^2)$, they are identical algorithms under hyperparameter tuning. We label this merged baseline squared L_2 / KL. It makes every state a restoring force strictly proportional to its drift.
- **Sub-quadratic / Saturating (H^2, TV):** $g(x) = 1 - e^{-x^2/\tau}$, exactly matching H^2 ($\tau = 8\sigma^2$). Both H^2 and TV plateau at an upper bound of 1, effectively releasing far-drifted outlier states. We use smooth H^2, which is merely a monotonic reparametrization yielding redundant empirical behavior.
- **Super-quadratic / Threshold (χ^2, Trust region):** $g(x) = \mathrm{relu}(x - \epsilon)^2$. While χ^2 represents the super-quadratic class, its exponential term e^{d^2/σ^2} produces unbounded gradients that cause numerical instability, conflating optimization failure with penalty shape. We instead instantiate this regime via a hinge trust region ($\epsilon = \sigma\sqrt{2\delta}$), cleanly capturing the TRPO (Schulman et al., 2015) principle of "zero penalty within a safety margin, quadratic penalty beyond it" with well-behaved curvature.

(ii) Underlying metric M.

- **Function-space Fisher metric (default):** For fixed σ, the Fisher information is $F = G/\sigma^2$, where G is the Gauss–Newton matrix of policy outputs. This restricts updates strictly along directions that alter the output action distribution (Hinton et al., 2015; Li & Hoiem, 2017).
- **Parameter-space L_2:** $c\,\|\theta - \theta_s\|_2^2$, where θ_s is π_s's parameter. This replaces F with the isotropic identity metric I. Because policy representations contain substantial null spaces that do not affect output actions, an unweighted parameter penalty uniformly penalizes all directions, incurring substantial cloning overhead for the same behavioral proximity (Kirkpatrick et al., 2017).
- **Weight decay:** Standard Adam weight decay on the policy head shrinks weights toward $\theta = 0$ rather than θ_s. This isolates generic capacity control from explicit behavioral anchoring.

(iii) The state distribution $\mathcal{D}$. The deployed setting trains the policy over all states in $\mathcal{D}_s^d$, matching the training distribution of the world model. The variant restricts evaluations to the expert-and-success subset $\mathcal{D}_s^e$ on which π_s was cloned, testing whether regularization must cover the full dynamics support or only the task manifold.

(iv) The reference target π_{ref}. The deployed setting pulls towards the frozen policy $\pi_s(s)$ evaluated at s. The variants replace evaluations with the static actions stored in dataset transitions over

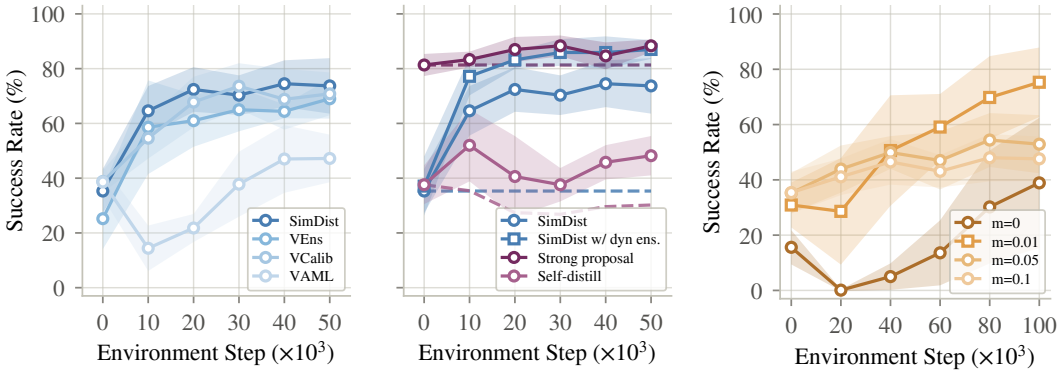

Figure 7: **More experiments on improving value estimating and proposal policy.** Left: SimDist compared with value-aware model learning (VAML), a pessimistic value ensemble (VEns), and value calibration (VCalib). Right: proposal self-distillation and a stronger proposal trained on 500 target demonstrations, compared with SimDist with and without a dynamics ensemble using 20 demonstrations. Solid curves show planner success; dashed curves show proposal-only success.

Figure 8: **Margin sweep in Dyna-IBRL.** We compare Dyna-IBRL with prior-preference margins $m \in \{0, 0.01, 0.05, 0.1\}$ over 100k target interactions. The margin sets how much the learned actor must outperform the prior before it is selected over the prior.

either all of $\mathcal{D}_s^d$ or the expert-and-success subset $\mathcal{D}_s^e$. This tests whether interactive query access to π_s is necessary or if offline dataset labels suffice.

To fairly compare these variants, we calibrate λ to make sure the coefficient provides comparable strength for them. Before the adaptation, we train the policy with different variants and sweep their λ until it matches the default setting (Squared L_2 / KL, $\lambda = 1$) on

$$D_{\text{src}} = \frac{1}{d_a} \mathbb{E}_{s \sim \mathcal{D}_s^d} \left\| \pi_\theta(s) - \pi_s(s) \right\|_1, \tag{13}$$

with d_a the action dimension. Adaptation is only run when D_{src} approximately matches the default, i.e., 0.025. This results in a wide range of final strengths shown in Tab. 4. Weight decay and the unfiltered recorded actions cannot reach the working point at any strength and are reported at their attainable floor.

D Supplementary Experiments

D.1 Failed attempts

In addition to dynamics ensembles and policy tethering, we explore several other ways to improve SimDist. Since planning in the sparse-reward setting relies primarily on the value function for decision making, an alternative to reducing dynamics-model error is to directly reduce value-estimation error, so that the value function can make more accurate predictions on imagined latent features produced by imperfect rollouts.

Specifically, we investigate: (i) adapting the dynamics model through value estimation, namely Value-Aware Model Learning (VAML); (ii) penalizing value uncertainty using a pessimistic value ensemble; and (iii) reducing value-estimation error on imagined latent features through value calibration in imagined states. We also explored several improvements to the policy: (iv) training the proposal policy to imitate successful MPPI trajectories, thereby further improving proposal quality (Self-Distill); and (v) fine-tuning the proposal policy with additional demonstrations before adaptation (Strong Proposal).

However, none of these attempts lead to effective adaptation. These results shown in Fig. 7 suggest that the main performance bottleneck of model-based planning lies in the accuracy of the learned dynamics model, rather than in value estimation or candidate-action generation.

(i) Value-aware model learning (VAML). Since the planner only reads the dynamics through $\hat{V}_\theta$, errors of f_θ that leave the value unchanged are harmless to the ranking. Value-aware model learning (Farahmand et al., 2017; Farahmand, 2018; Voelcker et al., 2025) therefore penalizes only the part of the model error that the value can see. We replace the dynamics training loss with

$$\mathcal{L}_{\text{VAML}}(f_\theta) \;=\; \mathbb{E}_{\mathcal{R}}\Big[\tfrac{1}{\sum_k \delta^k}\sum_{k=0}^{H-1}\delta^k\,\big(\hat{V}_\theta(\hat{z}_{t+k+1}) - \operatorname{sg}\hat{V}_\theta\big(E_\theta(s_{t+k+1})\big)\big)^2\Big], \tag{14}$$

with $\hat{V}_\theta$ frozen, so gradients reach f_θ only through the value. Although VAML reduces value-estimation error, it increases the distance between the terminal latent state imagined by the model $\hat{z}_{t+H}$ and the terminal latent state obtained from actual environment rollouts z_{t+H}. This ultimately degrades the latent geometry on which planning relies to rank trajectories according to their value estimates.

(ii) Value ensemble with pessimism (VEns). Following MRS.Q (Chang et al., 2026), we replace the value head by an ensemble of $N = 10$ heads, each on an independent Bernoulli(0.5) mask m_i of every batch and with a frozen random prior function p_i of scale $\beta = 3$ to keep the members diverse:

$$\hat{V}_i(z) \;=\; g_i(z) + \beta\, p_i(z), \qquad \mathcal{L}_V \;=\; \tfrac{1}{N}\sum_{i=1}^{N}\mathbb{E}\big[m_i\,\ell\big(\hat{V}_i(z_t),\, Q^e_s(s_t, \pi^e_s(s_t))\big)\big], \tag{15}$$

and the planner scores the terminal state pessimistically with $\hat{V}_{\min}(z) = \min_i \hat{V}_i(z)$ in place of $\hat{V}_\theta$. However, this pessimistic value-ensemble approach is effective only when the ensemble members disagree in their value estimates. In our setting, the dynamics gap changes $p(s' \mid s, a)$ but does not change the marginal state distribution $p(s)$. The level of disagreement on states from target-domain trajectories is nearly identical to that on corresponding states from the source domain (0.0083 v.s. 0.0078), indicating that this method is ineffective during adaptation.

(iii) Reducing the value error at the imagined state (VCalib). The planner evaluates $\hat{V}_\theta$ at the imagined latent $\hat{z}_{t+H}$, not at the true one z_{t+H}, so its effective value error is $\varepsilon(s_t, a_{t:t+H}) = \hat{V}_\theta(\hat{z}_{t+H}) - \hat{V}_\theta(z_{t+H})$. In the oracle dynamics experiment, feeding the real z_{t+H} to the same value head solves the task. Therefore, it is natural to calibrate the value to the imagined latent and remove ε. Each round, after the dynamics update, we regress the value head on real target segments from $\mathcal{R}$:

$$\min_{\hat{V}_\theta}\; \mathbb{E}_{(s_t,\, a_{t:t+H},\, s_{t+H})\sim\mathcal{R}}\Big[\ell\Big(\hat{V}_\theta\big(f^H_\theta(E_\theta(s_t), a_{t:t+H})\big),\; \hat{V}^0_\theta\big(E_\theta(s_{t+H})\big)\Big)\Big], \tag{16}$$

where f^H_θ is the H-step latent rollout (no gradient) and $\hat{V}^0_\theta$ is a frozen copy of the source value head. The imagined-state error does go down, but the planner does not improve. This is because MPPI relies on value-based ranking, meaning that it is the relative value error across candidate trajectories that significantly affects MPPI. Value calibration helps reduce the absolute value error, but not the relative error between candidate trajectories.

(iv) Self-distilling the planner into the proposal policy (Self-distill). The planner is a policy-improvement operator on top of π_θ, so its successful rollouts could serve as new demonstrations. Let $\mathcal{R}^+ \subset \mathcal{R}$ be the transitions of successful target episodes, whose actions are the MPPI outputs. After the first 5k transitions, each round trains on

$$\min_{\pi_\theta}\; \tfrac{1}{2}\,\mathbb{E}_{(s,a)\sim\mathcal{D}_t}\big\|\pi_\theta(\operatorname{sg}(z)) - a\big\|_2^2 \;+\; \tfrac{1}{2}\,\mathbb{E}_{(s,a)\sim\mathcal{R}^+}\big\|\pi_\theta(\operatorname{sg}(z)) - a\big\|_2^2, \tag{17}$$

where half of each batch drawn from $\mathcal{D}_t$ keeps the proposal policy from collapsing onto a few rollouts. The loop only clones successes, which already come from where the planner works well. However, this self-distillation strategy does not effectively improve the proposal policy. The dashed curve in the right panel of Fig. 7 shows that the degraded performance of the proposal policy itself also negatively affects MPPI. To investigate whether a sufficiently strong proposal policy can improve adaptation, we conduct experiment (v).

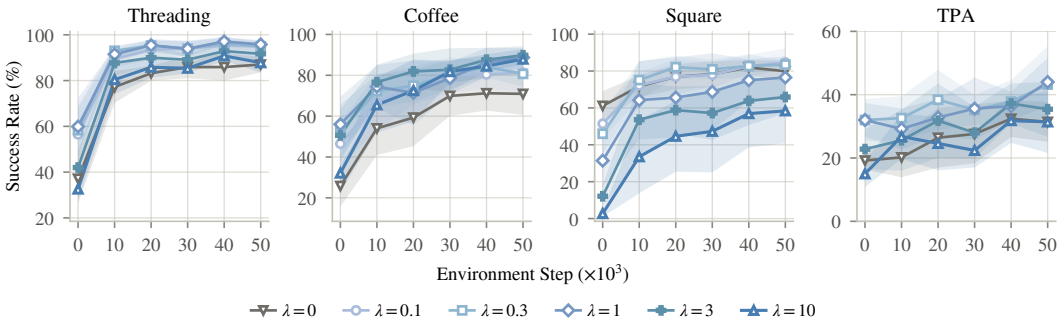

Figure 9: **Lambda sweep on 4 manipulation tasks.** We sweep $\lambda \in \{0.1, 0.3, 1, 3, 10\}$ during proposal warm-up on Threading, Coffee, Square, and TPA, then evaluate adaptation over 50k target interactions. Larger λ penalizes deviations from the source policy more strongly.

(v) More target demonstrations for the proposal policy (Strong proposal). If the proposal policy were the bottleneck, a stronger proposal policy should raise the planner. We keep all other settings fixed and increase the target demonstration set from 20 to 500 episodes:

$$\min_{\pi_\theta} \mathbb{E}_{(s,a)\sim\mathcal{D}_t^{(500)}} \left\| \pi_\theta\big(\mathrm{sg}(E_\theta(s))\big) - a \right\|_2^2 \qquad \text{instead of} \qquad \mathcal{D}_t^{(20)}. \tag{18}$$

The number of BC steps is raised from 1k to 8k so that the larger set is actually fitted. The deep red dashed line in Fig. 7 shows that the proposal policy does become stronger. Even so, SimDist with a dynamics ensemble matches its performance after adaptation with only 30k transitions. The former uses only 20 demonstrations and improves the proposal policy from an initial success rate of 38% to 85% through MPPI, while the latter improves an already strong proposal policy from 80% to only 85%. This suggests that fully realizing the potential of model-based planning does not depend on how strong the proposal policy is, but rather on the accuracy of the dynamics model.

D.2 Sweep experiments

Margin. After sweeping m, we find $m = 0.01$ is the best for IBRL adaptation, as it effectively uses the prior to warm up. As depicted in Fig. 8, when m is reduced (like $m = 0$), the actor wins the argmax immediately, before the critic can meaningfully rank either candidate, and the policy needs more budget to learn to be optimal. When m increases (like $m = 0.1$), the actor never wins the argmax at any logging point, so the prior carries over every episode and little adaptation occurs, and policy performance improves slowly.

Tether strength. We analyze $\lambda \in \{0.1, 0.3, 1, 3, 10\}$ across four tasks and report the best-performing setting for each task in Fig. 4. Illustrated in Fig. 9, we find that the optimal λ varies across tasks. For manipulation tasks, the best values are mostly concentrated between 0.1 and 1, suggesting that when target-domain demonstrations are limited, a relatively weak tether often performs better. It also shows that the tether strength introduces a clear trade-off: a tether that is too weak, or no tether at all, performs worse, while a tether that is too strong, or directly fine-tuning on target-domain demonstrations, is also inferior to choosing an appropriate tether strength.

Exploration optimism, Planning penalty and Dynamics ensemble number. We test whether dynamics ensemble disagreement can guide exploration or improve planning. Let $J(\mathbf{a})$ denote the original planner score for a candidate action sequence $\mathbf{a}$, and let $U(\mathbf{a})$ denote its accumulated dynamics ensemble disagreement. Optimistic exploration rewards uncertain trajectories, while pessimistic planning penalizes them:

$$J_{\text{explore}}(\mathbf{a}) = J(\mathbf{a}) + c_{\text{explore}} U(\mathbf{a}), \qquad J_{\text{plan}}(\mathbf{a}) = J(\mathbf{a}) - \beta_{\text{plan}} U(\mathbf{a}). \tag{19}$$

The exploration bonus is used during target-data collection, while the planning penalty is applied when scoring candidate trajectories. Fig. 10 compares these choices on Threading. Neither provides a clear improvement over setting its coefficient to zero. Increasing the dynamics ensemble from one to five heads improves success, while ten heads provide no further gain. We therefore use five dynamics heads without exploration bonuses or planning penalties.

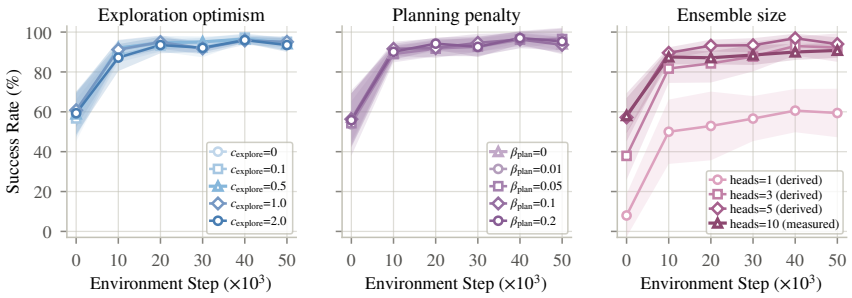

Figure 10: **Dynamics ensemble hyperparameters.** On Threading, we vary the exploration bonus coefficient c_{explore} (left), the planning penalty coefficient β_{plan} (middle), and the number of dynamics heads (right). The first two use ensemble disagreement to encourage exploration or penalize uncertain plans, respectively. All panels report success over 50k target interactions.

Table 12: **Candidate selection with adapted dynamics models.** Ranking metrics use target-domain candidates near the proposal policy. The variants use one, three or five heads from different adapted checkpoint, keeping the rest unchanged.

Dynamics head number	Rank $\delta \uparrow$	Regret$/\sigma \downarrow$
1	0.532	0.811
3	0.566	0.724
5	0.595	0.706

D.3 Dynamics ensembles and candidate selection

We evaluate whether dynamics ensembles improve candidate selection in the target domain. We consider candidate action sequences near the proposal policy and compare their scores after model rollouts with their scores after true environment rollouts. Each model uses its own value head for both scores:

$$\hat{v}_i = \hat{V}(\hat{z}_H^i), \qquad v_i = \hat{V}(E(s_H^i)), \tag{20}$$

where $\hat{z}_H^i$ is the imagined terminal latent and s_H^i is the true terminal state for candidate i. We report Spearman rank correlation δ between $\{\hat{v}_i\}$ and $\{v_i\}$, and normalized selection regret:

$$\frac{\text{regret}}{\sigma} = \frac{\max_i v_i - v_{\hat{i}}}{\text{Std}_i(v_i)}, \qquad \hat{i} = \arg\max_i \hat{v}_i. \tag{21}$$

Higher correlation and lower regret indicate better candidate selection. The reference scores v_i are value estimates at true terminal states, not observed returns. These metrics measure how model rollouts affect candidate scores and they do not measure the accuracy of the value head itself.

Shown in Fig. 12, using one, three, and five heads progressively improves both rank and regret, with smaller gains from three to five heads. This supports averaging multiple dynamics predictions to improve candidate selection. We limit the conclusion to candidates near the proposal policy. The ensemble still has substantial selection error, and these results do not imply oracle-level predictions.

D.4 Visual experiment

We test DomainADEPT and SimDist in pixel-based environment: Threading. As shown in Fig. 11, DomainADEPT consistently outperforms SimDist throughout training. DomainADEPT reaches a success rate of approximately 34% after only 30k environment steps, exceeding the approximately 31% achieved by SimDist at 100k steps. Its success rate peaks at approximately 46% at 90k steps and remains higher than that of SimDist at the end of training (41% versus 31%). These results demonstrate that DomainADEPT achieves stronger performance with fewer environment interactions.

D.5 Preliminary real-robot grasping results

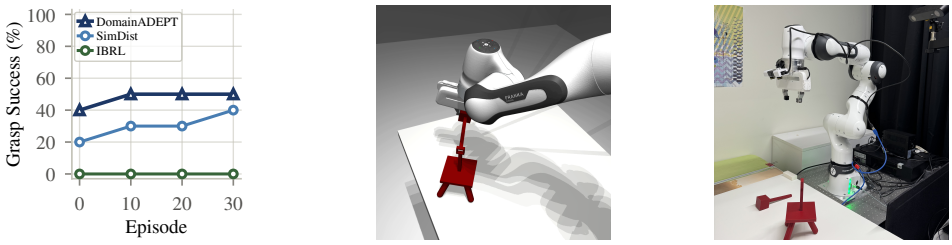

Figure 12: **Threading on the real robot.** *Left:* grasp success rate against the number of target episodes consumed. *Middle:* the task in simulation. *Right:* the same task on the real robot. The policy is updated every 10 episodes, and each point is evaluated over 10 trials.

To evaluate whether DOMAINADEPT transfers effectively to a real robot, we conduct experiments on a physical Franka Research 3 robot. We use two ZED cameras: a global camera ZED 2i that captures scene-level information and a wrist camera ZED mini that provides local observations. The simulation environment is constructed in MuJoCo using a Real-to-Sim pipeline. We reproduce the real setup as closely as possible by matching the lighting conditions, camera viewpoints, object appearance, table size and color, while randomizing factors such as the background, illumination, and object positions to improve the robustness of visual learning. The goal is to minimize the visual sim-to-real gap as much as possible so that the experiments can focus primarily on the dynamics gap.

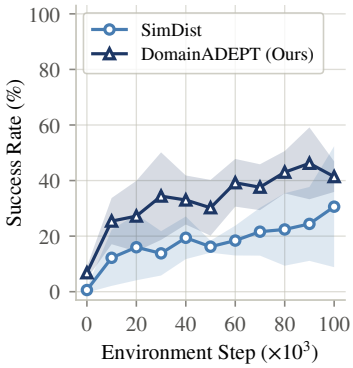

Figure 11: **Pixel experiment on Threading.** DOMAINADEPT consistently achieves higher success rates across the evaluated training steps, demonstrating improved sample efficiency. Shaded regions indicate variability.

We report grasp success rather than full-task completion. No full-task successes are observed in the evaluated trials, so these results provide only a preliminary assessment of the grasping stage. In particular, the remaining sim-to-real visual gap is not fully resolved, which can lead to errors in the learned visual representation and, consequently, inaccurate dynamics and value predictions during real-robot adaptation. This prevents the current system from completing the full task reliably. Since our study primarily focuses on adaptation to dynamics shifts rather than visual-domain adaptation, addressing this residual visual gap is left for future work.

E LIMITATION AND FUTURE WORK

First, our analysis focuses on adaptation to dynamics shifts under a limited interaction budget, primarily in sparse-reward manipulation tasks. The controlled cross-domain setting allows us to separate value transfer from world-model adaptation, but does not capture the full complexity of physical deployment, including visual discrepancies, sensor noise, and partial observability. Our preliminary real-robot experiments report grasping results, while full-task success has not yet been achieved. Extending DOMAINADEPT to effective end-to-end sim-to-real adaptation remains an important direction for future work.

Second, value updates perform poorly in our experiments. This result supports freezing the value function under our protocol, but it does not rule out the possibility that value adaptation may become effective with more data or different update strategies. Our method also assumes access to target-domain demonstrations for fine-tuning the proposal policy. Relaxing these assumptions, or dynamically adjusting the strength of source-domain regularization during adaptation, remains an important direction for future work.

Finally, planning and dynamics-model ensembles trade additional computation for improved interaction efficiency in the target domain. Our comparisons control for the target-domain interaction budget rather than the computational budget. Our diagnostic analysis of candidate-policy ranking shows that model ensembles improve planning-relevant predictions near the proposal policy, but this improvement does not necessarily hold in whole space. These results suggest that model accuracy should be evaluated in relation to the region searched by the planner, rather than treated as a single global metric.