Value Conditioned Policy Fine Tuning for Test Time Domain Adaptation

Harit Pandya [1] Ignas Budvytis [1] Rudra P. K. Poudel [1] Stephan Liwicki [1]

Abstract

Rapid cross-domain adaptation of learned policies is a key enabler for efficient robot deployment to new environments. Especially sim-to-real transfer remains a core challenge in reinforcement learning (RL), due to the unavoidable difference in world dynamics. While naïve policy updates with fine-tuning are unstable due to noisy gradients under domain shifts, other methods typically learn a new policy from scratch, relying on data points from the source and target domains using selective data sharing or reward shaping. However, neither approach is suitable for time-efficient policy adaptation or adaptation without access to an efficient simulator during deployment. On the other hand, we propose a value conditioned policy fine tuning that leverages the existing Q-function to estimate trust regions for a stable policy update. In practice, this can be achieved simply by combining gradients from the pre-trained and current Q-functions. We conduct extensive experiments on the MuJoCo dynamics adaptation benchmark for online adaptation, demonstrating competitive performance compared to existing state-of-the-art methods with over 3.5x times faster runtime.

1. Introduction

Building systems that learn to perform tasks autonomously has long been a goal in machine learning, robotics, computer vision, and natural language processing. Reinforcement learning (RL) has experienced immense success regarding its applicability to wide range of fields, evolving from solving simple Markov Decision Processes with well-defined state and action spaces (e.g., tabular RL (Howard, 1960)) to tackling complex, open-world tasks such as autonomous driving, intricate manipulation with world models (Guan et al., 2024), and even training advanced reasoning models such as Deepseek-R1 (Guo et al., 2025).

However, deploying RL efficiently in real-world scenarios requires rapid adaptation to unforeseen changes in the environment. For example, in object manipulation tasks, adaptation may be necessary due to changes in hardware structure (e.g., the number of fingers in a robotic gripper) or object properties (e.g., weight or friction). While adaptation to appearance variations has been widely studied (Finn et al., 2017; Yang et al., 2024; Hansen et al., 2020b), rapid adaptation to out-of-distribution physics remains an active area of research (Eysenbach et al., 2020; Xu et al., 2023; Lyu et al., 2024). Recent approaches frame cross-domain adaptation of dynamics and morphology as a hybrid RL problem, where policies are trained jointly on data from both the source and target domains. We refer the readers to the Appendix Section B for a further study of related works literature. However, these methods require extensive training before they can be effectively deployed in the target domain, limiting their real-world practicality.

Our approach, instead, is formulated as a trust-region-based constraint optimization objective, which aims to conservatively update the policy, by the agreement of gradients of source and target domain Q-functions. Initially, it prioritizes the source gradients when the target Q-function gradients are noisy, then gradually shifts more weight toward the target as learning progresses and the target gradients become more reliable. Hence, our method avoids the need of retraining from scratch, by applying soft constraints on the learning process (i.e. restricting the policy updates), that typically are needed to avoid noisy policy updates. We evaluate our approach on established dynamics variations and morphological adaptation benchmarks from (Lyu et al., 2024; Xu et al., 2023), and we show results for online and offline learning from the source domain. We compare our formulation with the state-of-the-art, such as reward shaping, importance sampling, filtering-based cross-adaptation approaches and fine tuning, and obtain competitive performance while having $3.5\times$ faster runtime for adaptation as compared to the state-of-the-art.

Our contributions can be summarized as follows: We formulate cross-domain fine-tuning of RL policies as a trust-region-based constrained optimization problem, enabling conservative policy updates that mitigate the impact of noisy gradients. Through extensive experiments across diverse environments with dynamic shifts in both offline and online settings, we demonstrate that fine-tuning across domains can

[1]Toshiba Europe Ltd.. Correspondence to: Harit Pandya <harit.pandya@toshiba.eu>.

achieve competitive performance while providing a $3.5\times$ speed-up compared to state-of-the-art approaches that require full policy retraining.

2. Preliminaries

The reinforcement learning (RL) problem can be formulated as a Markov Decision Process (MDP), which can be specified by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where S is the state space, $\mathcal{A}$ is the action space, P denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to R$ is the scalar reward signal, and $\gamma \in [0, 1)$ is the discount factor. The objective of RL is to find a policy $\pi(a|s; \phi)$ that maximize the discounted cumulative return $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

Similar to (Lyu et al., 2024), we define source domain by MDP $\mathcal{M}_{src} = (\mathcal{S}, \mathcal{A}, P_{src}, r, \gamma)$ and a target domain using MDP $\mathcal{M}_{\text{tar}} = (\mathcal{S}, \mathcal{A}, P_{tar}, r, \gamma)$, hence we assume that state space and action space is shared, while only transition dynamics differ. We further assume that rewards are bounded by a maximum $rmax$ such that $|r(s, a)| \leq rmax, \forall s, a \in \mathcal{S}, \mathcal{A}$. We assume access to an expert policy parameterized by (ϕ^*) pre-trained in source domain $(\pi^*_{src}(a|s; \phi^*))$ along with its action-value function $Q^*_{src}(s, a; \theta^*)$ parameterized by θ^* and replay buffer $\mathcal{D}_{src}$ are available. Following PAR (Lyu et al., 2024), we specify the normalized probability that a policy (π) encounters for state-action pair (s, a) in a domain $\mathcal{M}$ by $\rho_{\mathcal{M}}(s, a) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P^{\pi}_{\mathcal{M},t}(s_t) \pi(a_t|s_t)$, where $P^{\pi}_{\mathcal{M},t}(s_t)$ is the probability that the policy π encounters the state s at timestep t in the domain $\mathcal{M}$.

3. DAFT: Domain Adaptive Fine-Tuning

Previous methods (Niu et al., 2022; Xu et al., 2023; Lyu et al., 2024) formulate cross-domain adaptation as a max-entropy RL conditioned on assuming unlimited access to the source domain for co-training policy in source and target domain. This requires on-policy exploration in source domain to collect transitions(s, a, r, s'), from state s to next-state s' through action a obtaining reward r, that are similar to transitions in target domain. While such approaches are sample efficient in target domain they still require large amount of online training in source domain to collect similar transitions, resulting in much longer compute time for adaptation, which reduces their practicality especially in case of policy deployed in real-world. Furthermore, every time when a domain gap is encountered, re-training is required for adapting to the new domain. In our work, we ask: *can we formulate domain adaptation as fine tuning such that we can exploit the pre-trained policy without retraining from scratch?*

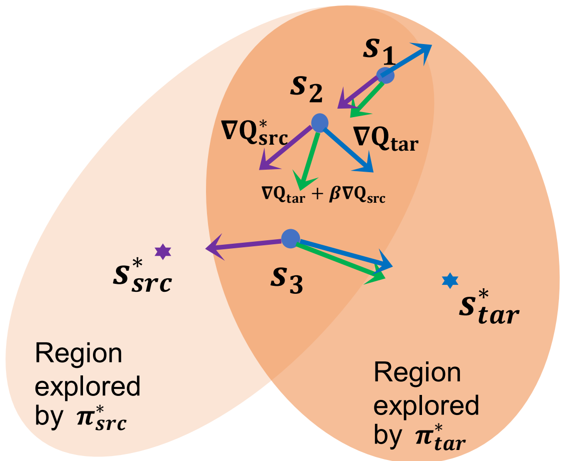
s_1
s_2
∇Q^*_{src}
∇Q_{tar}
$\nabla Q_{tar} + \beta \nabla Q_{src}$
s^*_{src}
s_3
s^*_{tar}
Region explored by π^*_{src}
Region explored by π^*_{tar}

Figure 1. Illustration of how KL-penalized optimization (Equation 2) corrects the gradient direction during fine tuning from domain $\mathcal{M}_{\text{src}}$ to $\mathcal{M}_{\text{tar}}$ with goal states as s^*_{src} and s^*_{tar}. Consider three states at s_1, s_2, s_3 collected by expert policy (π_{src}) in source domain. When the Q-function of policy is imperfect, for example, it is still learning, the KL-penalty β prefers the gradient direction ∇Q^*_{src} over ∇Q_{tar}. While after a few iterations, when the Q-function is confident, for instance, if $var(Q_{\text{tar}}(s_3))$ is low, KL-penalty β prefers the gradient direction ∇Q_{tar}.

3.1. Problem formulation

In case of naïvely implemented fine-tuning of actor-critic approaches in a target domain, which is different from source domain, previously learned policy and Q-functions pre-trained under source domain needs to be unlearned before learning the new policy under target domain. Since, the Q-function needs to be re-trained in target domain, and the initial updates of the Q-function are noisy, which in-turn could lead to generating out-of-training-distribution actions. As a result, large number of interactions are required to correct the behavior of the policy, that deteriorates sample efficiency.

Intuition: The core idea of our approach is that when the Q-function in the target domain (Q_{tar}) is noisy (still learning), the policy should take conservative actions that remain within the support of the expert Q-function (Q^*_{src}) from the source domain. To achieve this, we formulate domain-adaptive fine-tuning as a constrained optimization problem (see Equation 1), where the constraint ensures that the policy does not deviate significantly from the expert Q-function (Q^*_{src}) on source data. The KL trust region (ϵ) regulates the extent of deviation allowed from the expert Q-function (Q^*_{src}) and can be dynamically adjusted based on the confidence of the Q-function (Q_{tar}) in the target domain.

In practice, we enforce this constraint using a soft KL penalty (β) (see Equation 2), which simplifies the optimization by selecting β that balances the gradients of Q-functions from the source and target domains, as illustrated in Figure 1. This intuition can be formally expressed by modifying the maximum-entropy formulation of SAC (Haarnoja et al., 2018), with our changes highlighted in blue (Overall algorithm is presented in appendix section E).

$$\pi^*_{tar} = \arg\max_{\pi} \mathbb{E}_{a_t\sim\pi,\, s'\sim\rho^{\pi}_{\mathcal{M}_{\text{tar}}}} \left[\sum_t r(s_t, a_t) + \alpha \mathcal{H}\pi(\cdot|s_t) \right]$$
$$\text{s.t. } \mathbb{E}_{\rho^{\pi}_{\mathcal{M}_{\text{src}}}} \left[D_{\text{KL}} \left(\pi(\cdot|s') \,||\, \frac{exp(Q^*_{src}(s', \pi(\cdot|s')))}{Z^*_{src}(s', \pi(\cdot|s'))} \right) \right] \leq \epsilon. \quad (1)$$

Where, the $Q^*_{src}(\cdot,\cdot)$ is the action-value (Q) functions of expert policy (π^*_{src}) pre-trained in source domain, ϵ is the radius of the trust region and Z^*_{src} is a partition function that normalizes the $Q^*_{src}(\cdot,\cdot)$.

This formulation allows the functional behavior captured by Q-function of policy to fine tune gradually from expert policy in target domain by relaxing the trust region ϵ. Note that we condition the value function and not the behavioral policy itself i.e. we match the values and not state between source and target domain. This is advantageous when the dynamics gap between source and target domain is large. Moreover, the formulation does not require on-policy exploration in simulation assuming a pre-trained expert value function (Q^*) and replay buffer ($\mathcal{D}_{\text{src}}$) of data collected by expert policy is sufficient. Furthermore, a tight trust region initially (small ϵ) provides a strong bias towards adapting actions that generate rewards similar to that generated by expert policy. While, a relaxation of trust-region allows the value in the target domain to dominate, and the entropy encourages exploration in target domain. We base our approach on soft actor critic (Haarnoja et al., 2018) and specifically, we modify the policy update equation from (Haarnoja et al., 2018) by adding a KL-constraint from Equation 1 applied to the stochastic actor. Following SAC, we can rewrite the objective from Equation 1 in the form of simple multi-Q gradient using a KL-multiplier (β) which is a hyper-parameter as is gradually decayed over time based on exponential schedule. The resulting policy update rule can written as (changes from SAC are marked in blue):

$$\nabla_\phi J_\pi(\phi) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}_{\text{tar}}} [\nabla_\phi \log \pi_\phi(a_t \mid s_t) - \nabla_\phi Q_\theta(s_t, \pi_\phi(a_t|s_t))]$$
$$+ \beta \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}_{\text{src}}} [\nabla_\phi \log \pi_\phi(a_t \mid s_t)$$
$$- \nabla_\phi\, Q^*_{\theta^*}(s_t, \pi_\phi(a_t|s_t))]. \quad (2)$$

4. Experiments

For evaluating our approach when the source domain is online (i.e. training is performed using online RL). We follow the domain adaptation benchmark used by simulation based domain adaptation approaches (Lyu et al., 2024) and (Xu et al., 2023). Four Mujoco environments (half-cheetah, hopper, walker, ant) are selected as source domains from OpenAI Gym (Brockman, 2016). Target domain is simulated by modifying their dynamics directly, and indirectly through changed morphology creating eight evaluation scenarios, two for each environment. The dynamics are modified by limiting the rotation angle range of specific joints to simulate broken joint. While, morphology shifts are simulated by modifying the size of specific limbs ensuring the state/action space consistent across domains are retained.

We compare our approach (DAFT) to following simulator-based and fine-tuning-based approaches: (i) Simulator based: VGDF (Xu et al., 2023), a recent state-of-the-art method that filters transitions in the source domain that share similar value estimates as those in the target domain; PAR (Lyu et al., 2024) utilizes reward shaping by penalizing source domain rewards based on dynamics representation mis-match. Both these approaches co-train the policy from scratch on 10^6 steps in source domain (simulator) and 10^5 steps in target domain. (ii) Fine-tuning based: Naïve Fine-Tuning (FT) trains a SAC policy (Haarnoja et al., 2018) in source domain for 2 million steps and continues training in target domain for 10^5 steps. Simulator Guided Fine Tuning (SGFT) (Yin et al., 2024) is a concurrent work that uses reward shaping to add expected simulation reward to target reward. Similar to FT, we train SAC for 2 million steps in source domain and perform adaptation by training SGFT for 10^5 steps. We run all algorithms with the same four random seeds. The implementation details are given in Appendix (Section A). We do not compare with DARC (Eysenbach et al., 2020) since it has already been outperformed by PAR and VGDF.

The evaluation results are shown in Figure 2. It can be seen that our approach (DAFT) outperforms FT and SGFT in all the scenarios while being competitive to simulator based approaches (PAR and VGDF). Note, all three (PAR, DAFT and VGDF) are clear winner in 2 environments and are competitive in the other 2 environments. FT does not incorporate any difference when switching from source to target domain, thus due to initial noisy gradients, it needs to unlearn then again relearn the policy. On the contrary, SGFT adds estimated simulation rewards to existing target rewards which results in incorrect updates. While, PAR and VGDF are competitive, they require retraining for adaptation and hence take more time for training as shown in Figure 3. Where, the dynamics mis-match is smaller PAR wins since it is easier to match actions (representations of next

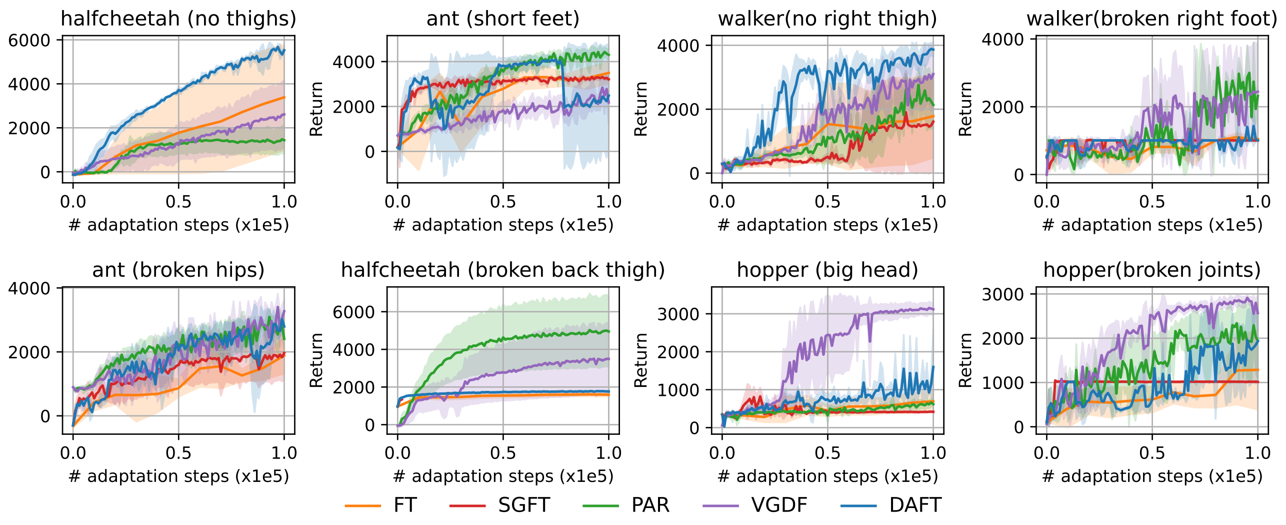

Figure 2. Adaptation performance comparison when the source domain is online. The curves depict the test performance of each algorithm in the target domain under kinematic shifts and morphology shifts. The modification to the environment is specified in the parentheses of the task name. The solid lines are the average returns over 4 different random seeds and the shaded region captures the standard deviation. DAFT shows superior performance over FT and VGFT in all environments, while remaining competitive with PAR and VGDF, each winning in 2 scenarios. Note that DAFT is significantly computationally efficient and does not require simulator at test-time adaptation.

states) on the contrary for larger dynamics mismatch it is crucial to consider the value function. DAFT works better in morphology adaptations where the horizon is limited. While, it suffers for longer horizon update tasks such as hopper, since pre-trained policy requires larger changes.

While we are competitive in terms of performance for adaptation, we possess significant advantages in terms of runtime of adaptation. We benchmark all on Intel(R) Xeon(R) Gold 6226R with single NVIDIA GeForce RTX 3090. Our approach takes 1.25 hours to finish 10^5 steps for half-cheetah, which is over $3.5\times$ faster as compared to PAR, over $8\times$ faster than VGDF and $1.5\times$ times slower than fine tuning. This shows that our approach is highly practical, efficient and no-match for simulator based approaches.

Further results for adaptation when the source domain is offline and ablation studies are available in the appendix sections D, C.

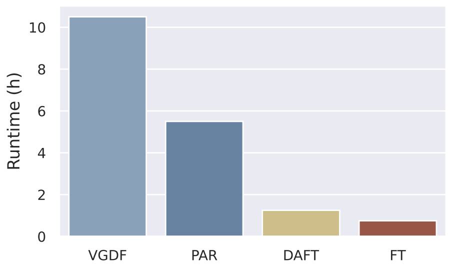

Figure 3. Runtime performance of different methods for 10^5 steps for adaptation. DAFT (ours) is over $3.5\times$ faster than PAR and over $8\times$ times faster than VGDF, while still achieving competitive performance.

5. Conclusion

We propose domain adaptive fine-tuning (DAFT) for cross-domain policy adaptation. Our approach provides an effective solution for rapid policy adaptation under domain shift with dynamics gap by fine-tuning a pre-trained policy without requiring retraining from scratch. By leveraging trust region updates, we successfully mitigate issues of noisy gradients and ensure stable policy adaptation. Our experiments on the MuJoCo dynamics adaptation benchmark with online and offline RL settings show that our proposed method outperforms existing state-of-the-art techniques, achieving competitive performance with a significantly faster runtime and no need for online access to a simulator. DAFT demonstrates high potential for more efficient and practical adaptation in real-world applications especially in those settings that require fast update speeds such as continuous control tasks.

Limitations and future work: Despite its effectiveness, DAFT has two main limitations: (i) DAFT currently lacks theoretical guarantees, which we leave for future work, and (ii) Real-world experiments are not included in this study; leaving sim-to-real transfer as an open challenge.

References

Brockman, G. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Eysenbach, B., Asawa, S., Chaudhari, S., Levine, S., and Salakhutdinov, R. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

Feng, Y., Hansen, N., Xiong, Z., Rajagopalan, C., and Wang, X. Finetuning offline world models in the real world. *arXiv preprint arXiv:2310.16029*, 2023.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pp. 357–368. PMLR, 2017.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Gamrian, S. and Goldberg, Y. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International conference on machine learning*, pp. 2063–2072. PMLR, 2019.

Guan, Y., Liao, H., Li, Z., Hu, J., Yuan, R., Li, Y., Zhang, G., and Xu, C. World models for autonomous driving: An initial survey. *arXiv preprint arXiv:2403.02622*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Hansen, N., Jangir, R., Sun, Y., Alenyà, G., Abbeel, P., Efros, A. A., Pinto, L., and Wang, X. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020a.

Hansen, N., Jangir, R., Sun, Y., Alenyà, G., Abbeel, P., Efros, A. A., Pinto, L., and Wang, X. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020b.

Howard, R. A. Dynamic programming and markov processes. *MIT Press google schola*, 2:39–47, 1960.

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Kumar, A., Fu, Z., Pathak, D., and Malik, J. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

Liu, J., Zhang, H., and Wang, D. Dara: Dynamics-aware reward augmentation in offline reinforcement learning. *arXiv preprint arXiv:2203.06662*, 2022a.

Liu, X., Pathak, D., and Kitani, K. M. Revolver: Continuous evolutionary models for robot-to-robot policy transfer. *arXiv preprint arXiv:2202.05244*, 2022b.

Lyu, J., Bai, C., Yang, J., Lu, Z., and Li, X. Cross-domain policy adaptation by capturing representation mismatch. *arXiv preprint arXiv:2405.15369*, 2024.

Mehta, B., Diaz, M., Golemo, F., Pal, C. J., and Paull, L. Active domain randomization. In *Conference on Robot Learning*, pp. 1162–1176. PMLR, 2020.

Memmel, M., Wagenmaker, A., Zhu, C., Yin, P., Fox, D., and Gupta, A. Asid: Active exploration for system identification in robotic manipulation. *arXiv preprint arXiv:2404.12308*, 2024.

Niu, H., Qiu, Y., Li, M., Zhou, G., Hu, J., Zhan, X., 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.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. 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.

Rafailov, R., Hatch, K. B., Kolev, V., Martin, J. D., Phielipp, M., and Finn, C. Moto: Offline pre-training to online fine-tuning for model-based robot learning. In *Conference on Robot Learning*, pp. 3654–3671. PMLR, 2023.

Smith, L., Kew, J. C., Peng, X. B., Ha, S., Tan, J., and Levine, S. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *2022*

International Conference on Robotics and Automation (ICRA), pp. 1593–1599. IEEE, 2022.

Wen, X., Bai, C., Xu, K., Yu, X., Zhang, Y., Li, X., and Wang, Z. Contrastive representation for data filtering in cross-domain offline reinforcement learning. *arXiv preprint arXiv:2405.06192*, 2024.

Xu, J., Makoviychuk, V., Narang, Y., Ramos, F., Matusik, W., Garg, A., and Macklin, M. Accelerated policy learning with parallel differentiable simulation. *arXiv preprint arXiv:2204.07137*, 2022.

Xu, K., Bai, C., Ma, X., Wang, D., Zhao, B., Wang, Z., Li, X., and Li, W. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 36:73395–73421, 2023.

Yang, S., Ze, Y., and Xu, H. Movie: Visual model-based policy adaptation for view generalization. *Advances in Neural Information Processing Systems*, 36, 2024.

Yin, P., Westenbroek, T., Bagaria, S., Huang, K., Cheng, C.-A., Kolobov, A., and Gupta, A. Rapidly adapting policies to the real-world via simulation-guided fine-tuning. In *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024. URL `https://openreview.net/forum?id=7KpnmGbCsK`.

Zhang, Q., Xiao, T., Efros, A. A., Pinto, L., and Wang, X. Learning cross-domain correspondence for control with dynamics cycle-consistency. *arXiv preprint arXiv:2012.09811*, 2020.

A. Implementation details

Environments: To achieve a fair benchmark we use the environments provided by (Xu et al., 2023; Lyu et al., 2024) without any modification. We refer readers to PAR (Lyu et al., 2024) for additional details and visualization of the adaptation environments. We additionally create two environments to evaluate large dynamics changes and motivate the use of Value function instead of state matching like PAR. Here, we modify the environment by inverting the action applied before performing any step. This simulates incorrectly connected joint motor. For offline settings we use D4RL benchmark and use medium dataset as proposed by (Xu et al., 2023).

Baselines: Our primary competitors are PAR (Lyu et al., 2024), VGDF (Xu et al., 2023), FT and SGFT (Yin et al., 2024). For VGDF and PAR we utilize the code available from official github repository and run the code. We observe that VGDF results are reproducible, while for PAR authors incorrectly evaluate on training environment in their code. Specifically, in the main file they generate evaluation environment using deepcopy from source environment. In case of morphological adaptations environment, the gym API ignores the custom XML file which creates a target environment similar to source environment. For SGFT, the code is not released yet however their approach is straightforward to implement.

Hyper-parameters: Next we mention the hyper-parameter used for running the evaluation. For online settings we train SAC in source environment for 2 million steps. We use the default hyper parameters from official repository. For SGFT, FT and DAFT, the adaptation policy is exact copy of the original policy, we keep rest of the hyper-parameters such as learning rate, same across the three approaches. For offline training we use CQL (Kumar et al., 2020) as baseline policy. This is used to extract q-function from offline dataset compensating for Q-overestimation.

In table A, we iterate the hyper-parameters used by us DAFT and SGFT and FT. For PAR and VGDF, we precisely use the hyper-parameters proposed by the authors in corresponding paper.

Table 1. Hyper-parameter values used in experiments for (DAFT,FT,SGFT).

Hyper-parameter	**Value**
online experiments	
Actor network	(256, 256)
Critic network	(256, 256)
Batch size	128
Learning rate	3×10^{-4}
Optimizer	Adam
Discount factor	0.99
Replay buffer size	10^6
Warmup steps	10^4
Nonlinearity	ReLU
Target update rate	5×10^{-3}
Temperature coefficient	0.2
Maximum log std	2
Minimum log std	-20
kappa	0.99 only used for DAFT
offline experiments	
policy eval start	40000
num epochs (pretraining for CQL)	2000
layer size	256
max path length	1000
batch size	256 (for pretraining), 128 for adaptation

B. Related work

Domain Adaptation in RL

Reinforcement learning policies can learn to accomplish tasks with minimal supervision but are highly sample inefficient. To address this, numerous approaches have been developed for rapid online adaptation, enabling policies to be pre-trained in simulation and transferred to the real world. However, the inherent domain gap between simulation and reality remains a fundamental challenge. Several factors contributing to this gap have been studied in the literature, including discrepancies in state space and perception (Gamrian & Goldberg, 2019; Hansen et al., 2020a), variations in system dynamics (Eysenbach et al., 2020; Liu et al., 2022a; Xu et al., 2023; Niu et al., 2022; Lyu et al., 2024), and differences in the action space of the embodiment (Liu et al., 2022b; Zhang et al., 2020). In this work, we specifically focus on the gap arising from differences in dynamics and state transitions.

Domain randomization was the focus of early works that try to learn a policy robust to domain shifts (Mehta et al., 2020; Peng et al., 2018). Others have proposed to predict privileged information (e.g. dynamics parameters (Kumar et al., 2021)) of the simulator for faster adaptation. However, these approaches fail in the the out-of-training-distribution generalization. Some recent methods model the adaptation of dynamics as system identification (Memmel et al., 2024; Xu et al., 2022) or assume access to a differentiable simulator and regress for target dynamics parameters from rollouts (Freeman et al., 2021; Hu et al., 2019). Here, once the dynamic parameters are identified, policies can be learned in the simulator with matched target dynamics. We note however, having a differentiable simulator is not always possible in complex manipulation tasks.

(Eysenbach et al., 2020) investigate dynamics adaptation with limited online interaction in the target domain while assuming access to a source domain with sufficient data. Specifically, this setting leverages a source domain simulator, and several approaches have been proposed to address the domain gap (Liu et al., 2022a; Xu et al., 2023; Niu et al., 2022; Lyu et al., 2024). These methods mitigate domain discrepancies through reward shaping (Eysenbach et al., 2020; Liu et al., 2022a; Lyu et al., 2024), importance re-weighting of the action-value function (Niu et al., 2022), or selective augmentation of simulator data from the source domain (Xu et al., 2023; Wen et al., 2024). DARC (Eysenbach et al., 2020) employs domain classifiers to quantify domain discrepancy, using this measure to penalize rewards obtained from the source domain. (Liu et al., 2022a) extend this idea to an offline setting, while PAR (Lyu et al., 2024) penalizes rewards based on representation mismatch. H2O (Niu et al., 2022) applies importance re-weighting, leveraging domain classifiers to refine action-value estimation. Additionally, (Xu et al., 2023) selectively transfers transitions from the source domain by evaluating differences in the value function.

Offline RL and Fine-tuning in the context of RL
Simulators serve as cost-effective proxies for training sample-inefficient reinforcement learning policies. A common approach involves fine-tuning a pre-trained simulation policy in the real world (Smith et al., 2022). In contrast, offline reinforcement learning methods leverage experience replay datasets to train policies entirely offline, without simulation (Kumar et al., 2020; Kostrikov et al., 2021). These methods demonstrate real-world policy deployment by initializing policies and Q-functions from offline data and continuing training with standard reinforcement learning techniques. However, they do not account for discrepancies between the source and target domains. Recent work has explored fine-tuning in model-based offline reinforcement learning (Feng et al., 2023; Rafailov et al., 2023), but similarly does not address domain mismatch. The closest related work to ours is a concurrent approach (Yin et al., 2024), which employs a simulator-guided fine-tuning strategy for sim-to-real transfer by augmenting the reward with the expected simulator reward. However, like other fine-tuning methods, it does not explicitly handle domain mismatch, leading to suboptimal performance in the presence of domain gaps, as demonstrated by our experiments. In contrast, we show that conservative updates enable efficient fine-tuning across domain discrepancies.

C. Results with Offline Source Domain

We next compare our approach when the source domain is offline dataset, this is slightly dis-advantageous to our approach since the assumption of sufficient coverage of expert policy breaks. Following the benchmark proposed by (Xu et al., 2023), we utilize the D4RL medium datasets (where the dataset is the replay buffer of a medium policy) (Fu et al., 2020) of four environments (i.e., half-cheetah, walker, hopper, ant) for evaluation. Modifications in dynamics and morphology are performed to create 8 evaluation scenarios as similar to online evaluation 4.

As a baseline we consider similar to online section (i) co-training based approaches VGDF+BC and PAR+BC and H2O. while, VGDF+BC and PAR+BC require an additional behavior cloning term to compensate for overestimation of Q-function which arises in offline domain. This limits the performance bounded by offline policy when dynamics mis-match is large, and could lead to instability while learning policy. Furthermore, these approaches including H2O require retraining from scratch which again limits practicality. In terms of fine tuning we compare with CQL+FT, where an offline CQL policy (Kumar et al., 2020) is trained in source domain for 200 epochs followed by fine-tuning using SAC in target domain for 10^5 steps. For fair comparison with CQL+FT, we also fine-tune DAFT from pre-trained CQL policy. Following (Lyu et al.,

Task Name	CQL-0	CQL+SAC	H2O	VGDF+BC	PAR+BC	DAFT
Half-cheetah (broken back thigh)	1128±156	3967±204	5450±194	4834±250	**5699±276**	5600±133
Half-cheetah (no thighs)	361±29	1184±211	2863±209	3910±160	1843±85	**4482±96**
Hopper (broken joints)	155±19	498±73	2467±323	**2785±75**	2011±1306	1435±953
Hopper (big head)	399±5	496±53	1451±480	**3060±60**	495±104	480±160
Walker (broken right foot)	1453±412	1877±1040	3309±418	3000±388	**3811±122**	3324±462
Walker (no right thigh)	975±131	1262±363	2225±546	**3293±306**	2617±842	**3475±415**
Ant (broken hips)	1230±99	-1814±431	2704±253	1713±366	**3303±175**	2177±1232
Ant (short feet)	1839±137	-807±255	3892±85	3120±469	**4925±138**	1518±885

Table 2. Performance comparison when the source domain is offline, i.e., only static source domain datasets are available. We report results (episode returns) on medium dataset from D4RL (Fu et al., 2020). The results are averaged over 4 varied random seeds. For DAFT, we train CQL from the dataset to obtain the Q-function which is fine-tuned in target domain. Note this more challenging for DAFT since, the baseline policy is not expert and state space is less explored. Still we are competitive with VDGF and PAR and clearly outperform fine tuning. Furthermore, VGDF and PAR require additional behavioral cloning (BC) to adjust for overestimation of Q-function in offline RL.

2024) we further add CQL-0 as baseline, where CQL evaluated in target domain in zero-shot manner. Note, we train PAR and SGFT, other results are taken from PAR. PAR and DAFT are trained in target domain for 10^5 time-steps.

We present the evaluation results in Table 2. It can be seen that naïve fine-tuning from an offline policy performs significantly worse, while DAFT using conservative updates shows competitive performance with PAR and VGDF even though the assumption of well explored expert policy is invalidated. Similar to online evaluation, here again we can observe that for the environments that require long horizon knowledge such hopper, the performance of DAFT is limited. Nevertheless, for morphological changes, DAFT outperforms other baselines. Overall, DAFT shows competitive performance being faster.

D. Ablation studies

Multi-objective vs reward shaping vs single critic
In this section we compare our proposed formulation of learning stochastic actor from (Equation 1) with learning a single critic (Equation 3) and reward shaping (where the source reward is penalized by difference in Q-function of source and target, i.e. $r = r - \|Q_{\text{tar}}(s,a) - Q^*_{\text{src}}(s,a)\|)\forall(s,a,r) \in D_{\text{tar}}$. For DAFT and single critic formulation we use same β-schedule. The comparison results are shown in Figure 4. Note that, we consider fine-tuning setting (and not retraining from scratch) similar to the online evaluation (Section 4), where all the approaches use same expert Q-function ($Q^*_{\text{src}}(s,a)$) and are initialized from same expert SAC policy trained in source environment for 2 million steps. It can be seen that our proposed formulation outperforms the other formulations in all the environments, which justifies the selection of the proposed stochastic actor formulation.

Parameter impact study As mentioned in Section 3, for implementation of the approach, we select a horizon (H) and compute an averaged gradients of Q^*_{src} for this horizon, which stabilizes the gradient for few tasks such as walker and helps in faster adaptation of the policy, on other tasks it does not show much influence. Here we select two environment, walker and half-cheetah and select horizon. It can be seen from Figure 5 that for half-cheetah the effect of horizon is negligible, while for walker, a longer horizon length=20 performs best. We consider horizon length as tuning parameter and empirically select 5 for all scenarios.

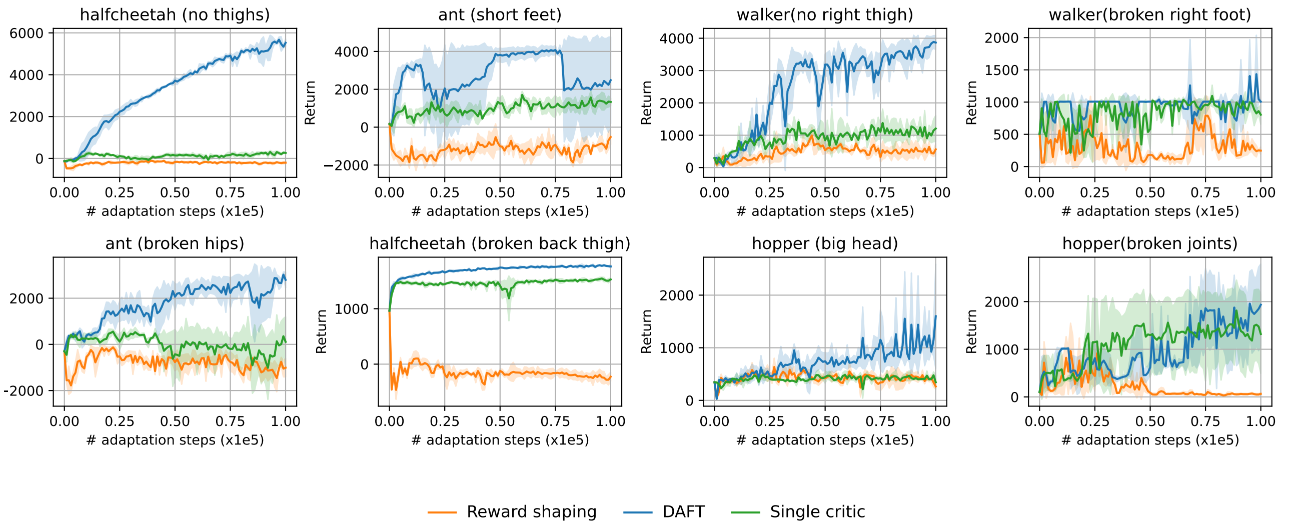

Figure 4. Comparison of alternative formulations, DAFT vs single critic vs reward shaping for cross-domain fine tuning of SAC policy where the source domain is online. All approaches are fine-tuned for 10^5 steps. The stochastic actor formulation (Equation 2) outperforms single critic (Equation 3) and reward reshaping in all environments.

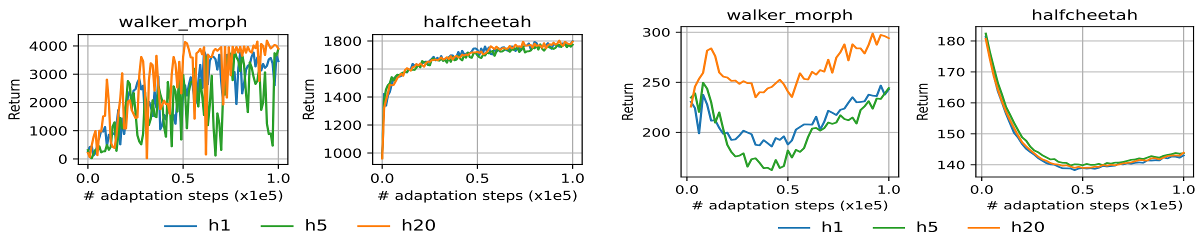

Figure 5. Effect of changing horizon on Q-values and episode returns. (left) Return values in the environment and (right) Q-values for different horizons (h=1,5,20). For fewer tasks like walker, longer horizon reduces the variance while for other task such as half-cheetah it does not impact.

E. Overall Algorithm

Algorithm 1 Domain Adaptive Fine Tuning (DAFT)

```
Input: Pretrained policy $\pi^*_{\text{src}}$, expert action-value function $Q^*_{\text{src}}$, and replay buffer $\mathcal{D}_{\text{src}}$
Output: Fine-tuned policy in target domain $\pi_{\text{tar}}$
Initialize policy from expert $\pi_{\text{tar}} \leftarrow \pi^*_{\text{src}}$, ensemble of Q-functions $Q^i \leftarrow Q^{\text{src}} + \mathcal{N}(0, \omega)$, and replay buffer $\mathcal{D}_{\text{tar}} \leftarrow \emptyset$
for each iteration $k$ do
    for time step $t = 1, \ldots, T$ do
        Rollout policy in target environment $\mathcal{M}_{\text{tar}}$ and collect transition $(s_{\text{tar}}, a_{\text{tar}}, r_{\text{tar}}, s'_{\text{tar}})$
        Add transition to target dataset $\mathcal{D}_{\text{tar}} \leftarrow \mathcal{D}_{\text{tar}} \cup \{(s_{\text{tar}}, a_{\text{tar}}, r_{\text{tar}}, s'_{\text{tar}})\}$
    Sample a mini-batch $\mathcal{B}_{\text{tar}}$ from $\mathcal{D}_{\text{tar}}$
    Sample a mini-batch $\mathcal{B}_{\text{src}}$ from $\mathcal{D}_{\text{src}}$
    Update target Q-functions using SAC policy iteration
    Estimate the $\beta$
    Update policy $\pi_{tar}$ using Equation 2;
```

Connection with VGDF and H2O

We applied the KL-penalty directly to the actor, where the actor tries to balance the gradients of the two critics (refer Equation 2). Alternatively, similar penalty can be applied to critic for learning a single critic, which can be written in the form of following Q-learning objective:

$$\begin{aligned} J_Q(\theta) &= \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{tar}}}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \hat{Q}_{\overline{\theta}}(s,a)\right)^2\right] \\ + \quad & \beta(s,a)\mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{sim}}}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \hat{Q}_{\overline{\theta}}(s,a)\right)^2\right]. \end{aligned} \tag{3}$$

The policy can then be learned simply applying SAC-policy-update and using Q from aforementioned objective. Here, we can estimate β from importance sampling using domain classifier, that results in H2O approach. Alternatively, we can estimate beta through a hard filter resulting in formulation proposed by VGDF. However, updating Q-function using Equation 3, for adaption can result in noisy or conflicting gradient updates resulting in sub-optimal performance as shown in our experiments in Section D Figure 4.