Adapting PAD to Trifinger Robotic Manipulation in Causal World

Alexandre St-Aubin

***Abstract*— Deep Reinforcement Learning (RL) policies frequently fail when deployed in environments that differ from their training conditions. In this work, we propose a self-supervised Test-Time Adaptation (TTA) framework that leverages an auxiliary Inverse Dynamics (ID) objective to update the agent's encoder during deployment. Unlike prior methods that update all encoder weights, we introduce an adaptation strategy that freezes the policy backbone and updates only the normalization statistics of the shared layers. We evaluate this method on the *CausalWorld* robotic benchmark under two distinct shift types: physical dynamics (robot mass) and spatial distribution (goal position). Our results show that joint training with the auxiliary task significantly improves baseline robustness. Furthermore, we find that Lifelong Adaptation outperforms episodic adaptation and zero-shot baselines, recovering policy performance on severe out-of-distribution (OOD) goal shifts where frozen agents fail. These findings suggest that constrained, self-supervised adaptation is a viable path for robustifying robot policies against unforeseen variations without access to reward signals.**

I. Introduction

Deep RL has demonstrated success in solving complex control tasks, ranging from video games to high-dimensional robotic manipulation. However, a critical limitation of standard RL algorithms is their resistance to distribution shifts. Policies trained in a fixed source environment often fail when deployed in a target environment that differs slightly in terms of visual appearance or physical dynamics. Bridging this generalization gap is essential for deploying RL agents in the real world, where non-stationarity and unmodeled perturbations are inevitable.

To address this, recent works have proposed TTA mechanisms that update the agent's parameters during deployment without access to reward signals. A prominent framework in this domain is *Self-Supervised Policy Adaptation during Deployment* (PAD) [1]. PAD leverages auxiliary self-supervised tasks, such as inverse dynamics prediction and rotation prediction, to fine-tune the policy's encoder on incoming observations. By aligning the representation of the target environment with the source environment, PAD has achieved state-of-the-art results on standard benchmarks like the DeepMind Control Suite.

However, the efficacy of PAD has only been demonstrated on relatively simple control tasks, such as 2D navigation and control (cartpole, walker) or single-arm reaching, where the domain shift is *visual*, specifically changing background colors. The framework relies on inductive biases tailored for image observations by using data augmentations like random cropping to learn more robust models. In contrast, the **CausalWorld** benchmark [2] presents a higher degree of complexity, involving complex tasks such as block pushing and pick-and-place that require coordinated control of a three-finger robotic hand. Learning these tasks directly from image observations is difficult due to scene complexity and occlusions, requiring multi-view setups to achieve adequate state estimation [3]. Therefore, we use *structured* (state-vector) observations to focus on the control challenge. CausalWorld allows for precise interventions on physical parameters, such as object mass, friction, joint gains, and goal positions, providing a great testbed to determine if the principles of PAD can be transferred from visual to *physical* and *structural* variations.

In this project, we investigate the adaptation of the PAD framework to this challenging robotic manipulation domain. Unlike the DeepMind Control Suite, CausalWorld involves causal interactions where the underlying dynamics of the simulation can be intervened upon. This work addresses two primary challenges:

1) **Modality Shift:** We transition the PAD framework from pixel-based encoders to state-based Multi-Layer Perceptrons (MLPs). This requires precise hyperparameter tuning and changes in the weight update logic at test time, detailed in section III.
2) **Complexity Gap:** We evaluate whether self-supervised objectives can capture meaningful features in a high-dimensional manipulation space, rather than the simple RL simulations evaluated in PAD.

II. Related Work

Here, we discuss in more depth the two building blocks of our approach, PAD and CausalWorld.

A. Self-Supervised Policy Adaptation during Deployment (PAD)

The PAD framework [1] operates on the premise that a robust policy requires representations that capture the underlying dynamics of the environment, invariant to superficial shifts. It employs a shared encoder architecture, denoted as π_e, which processes high-dimensional observations into a latent vector used by both the RL policy (Actor-Critic) and an auxiliary self-supervised task. During training, π_e is optimized jointly for maximizing reward and minimizing auxiliary loss. At test time, the RL objective is deactivated, and π_e is updated only via the auxiliary task to adapt to the target domain.

alxstaub@cs.ubc.ca

However, PAD has been validated primarily on the DeepMind Control Suite [4]. These tasks generally involve locomotion (e.g., Walker, Cheetah) in 2D planar environments or simple reaching tasks with a single robotic arm. The domain shifts introduced in these experiments are predominantly visual, such as replacing the background with a video, rather than physical.

B. CausalWorld and Robotic Manipulation Benchmarks

In contrast to the simple tasks used in adaptation literature, **CausalWorld** [2] provides a rigorous benchmark for contact-rich robotic manipulation. Based on the TriFinger platform [5], the environment simulates a manipulator with three fingers, each possessing three degrees of freedom, resulting in a nine-dimensional continuous action space. The task diversity in CausalWorld far exceeds simple reaching; agents must perform precise object manipulation, such as pushing specific blocks, pick-and-place operations, and constructing 3D towers under gravity.

Unlike 2D environments where the state is fully observable from a single viewpoint, the tasks in CausalWorld operate in a complex 3D workspace where self-occlusions by the robot fingers are frequent. To fully understand the scene from pixels, a setup involving three separate cameras is typically required, significantly increasing the difficulty of representation learning. Furthermore, CausalWorld explicitly models the causal structure of the environment, allowing for interventions on physical variables such as block mass, surface friction, link mass, and goal position. This allows us to evaluate adaptation not just against visual distractions, but against physical changes in the task. Our approach adapts the PAD framework to handle structured observations within the CausalWorld environment. We specifically modify the observation modality, the encoder architecture, and the test-time weight update logic.

C. Observation Modality

Initial experiments attempted to replicate the pixel-based workflow of the original PAD implementation within CausalWorld. However, learning robust policies from visual observations alone was too computationally expensive. The complexity of the TriFinger robot, combined with frequent self-occlusions, necessitates a multi-view setup (typically three cameras) to resolve the state ambiguity. Training a convolutional encoder on such high-dimensional inputs was not realistic.

Instead, we rely on low-dimensional structured state vectors s_t, which include things like the time left for the task, joint positions and velocities, end-effector positions, and information on the task's goal.

III. Method

A. Encoder Architecture

We replace the standard convolutional encoder π_e with an MLP π_l designed for structured observations. It features a projection layer and four hidden layers (128 units each) with layer normalization. We use a split architecture: the first two layers are shared, while the last two are independent. The logic is to separate perception from reasoning. The shared bottom layers learn a universal representation of the robot's physics valid for any task. The separated top layers allow the RL policy and the auxiliary model to process these features differently. If we shared the entire network, the distinct goals of maximizing reward and predicting dynamics would compete for the same feature space, likely degrading performance.

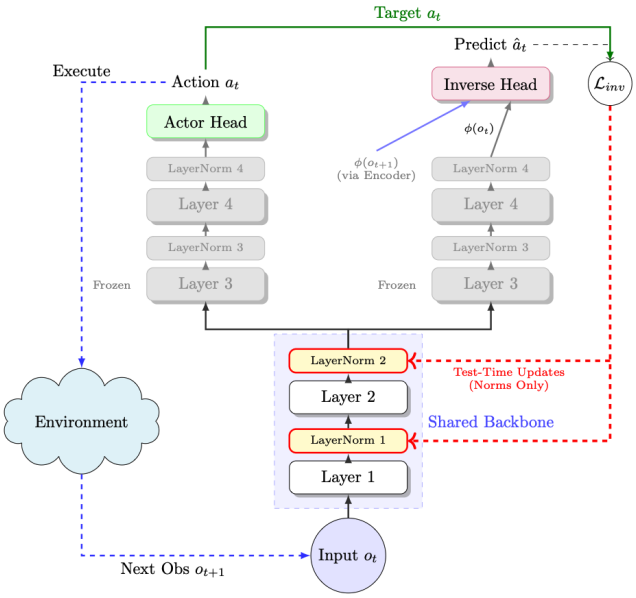

Fig. 1: The overall framework at test-time.

B. Self-Supervised Task Selection

The original PAD framework suggested rotation prediction and inverse dynamics prediction as its auxiliary tasks. Clearly, the first one is ill-defined for state vectors. Therefore, we exclusively utilize the **Inverse Dynamics** (ID) prediction task. The inverse dynamics head learns to predict the action a_t taken between two consecutive states s_t and s_{t+1}. This task forces the encoder to capture the underlying causal dynamics of the system, specifically, how control inputs influence state transitions.

C. Learning Framework

1) Training Phase: The learning objective during training is the joint optimization of the RL policy and the ID auxiliary task. The action a_t sampled from the RL policy at time t serves as the self-supervised target for the auxiliary task. Gradients are computed at each task-specific head and backpropagated through their respective independent layers. These gradients accumulate at the shared backbone (the first two layers), where the weights are updated to satisfy both objectives simultaneously. Throughout training, all simulation parameters (friction, arm link mass, goal position) remain fixed at their default values. No domain randomization is applied.

2) Test-Time Adaptation: During deployment, reward signals are unavailable, preventing RL policy updates. Adaptation relies solely on the self-supervised auxiliary task. Unlike the original PAD approach, which updates all shared encoder weights at test-time, we freeze the entire network with one exception: the normalization statistics. Specifically, we optimize only the affine parameters of the LayerNorm layers within the shared backbone using gradients from the inverse dynamics loss. We found that updating the full encoder weights at test-time led to instability and catastrophic forgetting, whereas adapting normalization statistics provided a more robust mechanism for aligning the input distribution without corrupting the learned representation. This framework is illustrated in Figure 1.

IV. Experiments

Our experiments aim to demonstrate the feasibility of self-supervised TTA for robotic manipulation tasks under physical shifts. Specifically, we investigate whether adapting only the normalization statistics of a shared encoder can recover policy performance in OOD environments.

A. Experimental Setup

We evaluate our method on the Reaching task within the *CausalWorld* benchmark. In this task, the robot must coordinate three robotic manipulators to reach specific 3D target coordinates. While CausalWorld supports complex object manipulation tasks (e.g., Pushing, Pick-and-Place), we utilize the Reaching task as a proof-of-concept. This primitive allows for rapid training cycles (≈ 50 epochs / 8 minutes).

B. Domain Randomization and Interventions

Unlike prior work in PAD which focused on visual domain randomization (e.g., video backgrounds), we introduce physical interventions to the simulation state. Initially, we hypothesized that increasing the mass of the robot links would serve as a good test of robustness. However, experiments revealed a ceiling effect. Models trained jointly with the auxiliary task (even without TTA) achieved near-perfect rewards across all mass scales tested. This shows that the auxiliary ID task forces the encoder to learn a representation robust to dynamics variations.

To rigorously stress-test the adaptation mechanism, we instead introduce increasingly severe shifts in *Goal Position*. The models are trained on a single, fixed goal configuration. At test time, we evaluate robustness by displacing the target goal radially outward from the original training goal position.

While ID is intuitively suited for dynamics adaptation (e.g., adapting to increased friction by predicting larger required forces), its utility for goal shifts is less obvious since the physics of the robot remain invariant. When the goal is moved to an unseen location, the observation vector (containing relative distances and target coordinates) shifts to an OOD region. We hypothesize that the self-supervised updates to the LayerNorm statistics will try to re-center these feature distributions, aligning the test-time observations with the training distribution that the frozen policy expects.

C. Baselines and Ablations

We compare the performance of the agent under four configurations to isolate the effects of the auxiliary task and the specific adaptation protocol.

1) **SAC (Baseline):** A standard Soft Actor-Critic agent trained only on the reward signal. It uses the same encoder architecture π_l but lacks the auxiliary inverse dynamics head. This serves as a lower bound to measure the contribution of the auxiliary task.
2) **Zero-Shot (No Adaptation):** The agent is trained jointly with the inverse dynamics auxiliary task, but the encoder is frozen during deployment. This isolates the benefit of the learned representation from the benefit of active test-time updates.
3) **Episodic Adaptation:** The full method, where the agent is trained jointly with the auxiliary task. At test time, the encoder's normalization statistics are updated via self-supervision within each episode, but the model is reset to its pre-trained state at the start of every new episode. This tests the method's ability to rapidly adapt from scratch.
4) **Lifelong Adaptation:** The full method, with a continuous learning setting. The encoder's normalization statistics are updated continuously across all test episodes without resetting. This tests the method's stability and ability to accumulate domain knowledge over time.

D. Evaluation Protocol

We report performance metrics averaged across multiple distinct evaluation trials to ensure statistical robustness. We define the difficulty of an evaluation task by the radial displacement (r) of the target goal from the training distribution. For a given difficulty scale r, we generate evaluation episodes by displacing the goal for each of the three robot arms. Specifically, the new goal for each arm is placed at a fixed distance r from its original training position, with an angle θ sampled uniformly from $[0, 2\pi)$.

For each method and difficulty scale, we execute 10 independent evaluation episodes. For each of these 10 episodes, the goal is exactly the same to allow the Lifelong Adaptation method to adapt to the new goal. To account for variance in model initialization, this entire procedure is repeated across 3 random seeds. The final reported metrics, **Average Episode Reward** and **Success Rate**, represent the mean performance across these $3 \times 10 = 30$ total trials per data point.

V. Results

In this section, we analyze the performance of the proposed adaptation strategies under increasingly severe distribution shifts. We report both Average Episode Reward and Success Rate across all methods.

A. Importance of the Auxiliary Task

The results in Table I highlight the critical role of the ID auxiliary task. The No Aux Task Baseline fails to solve the task even at the default training radius (0.0), achieving a

TABLE I: Comparison of **Average Episode Rewards** across increasing goal displacement radii.

	Average Episode Reward (↑)							
Method ↓ Radius →	**0.0**	**0.01**	**0.02**	**0.025**	**0.03**	**0.035**	**0.04**	**0.045**
No Aux Task (Baseline)	89.5	151.1	151.4	106.0	188.3	95.1	94.8	61.5
Zero-Shot (No Adaptation)	248.4	244.0	**219.2**	199.4	228.2	198.6	220.1	**167.6**
Episodic Adaptation	246.6	240.3	217.9	201.1	**229.6**	200.5	209.0	166.5
Lifelong Adaptation	**248.6**	**244.5**	218.9	**202.1**	227.1	**201.9**	**221.1**	164.2

TABLE II: Comparison of **Success Rate** across increasing goal displacement. Lifelong Adaptation achieves the highest success rates in the difficult $0.03\text{m} - 0.035\text{m}$ range.

	Success Rate (↑)							
Method ↓ Radius →	**0.0**	**0.01**	**0.02**	**0.025**	**0.03**	**0.035**	**0.04**	**0.045**
No Aux Task (Baseline)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Zero-Shot (No Adaptation)	100.0%	100.0%	100.0%	100.0%	10.0%	0.0%	0.0%	0.0%
Episodic Adaptation	100.0%	100.0%	100.0%	100.0%	40.0%	0.0%	0.0%	0.0%
Lifelong Adaptation	100.0%	100.0%	100.0%	100.0%	**60.0%**	**20.0%**	0.0%	0.0%

0% success rate and significantly lower rewards. In contrast, all methods using the joint training objective achieve near-perfect performance (100% success, > 240 reward) at the training distribution. This confirms that the auxiliary task forces the encoder to learn features that are essential for robotic control, regardless of TTA.

B. Adaptation in OOD Settings

We observe three distinct performance zones as the goal displacement radius increases. For small displacements ($0.0 - 0.025$), the learned representation is robust enough that adaptation is unnecessary. The Zero-Shot baseline maintains 100% success rate, and all methods perform comparably.

As the distribution shift becomes severe enough to break the frozen policy, we observe that at radius 0.03, the Zero-Shot method collapses to a 10% success rate. Episodic Adaptation recovers performance significantly (40% success), proving that test-time updates can realign the feature space. Lifelong Adaptation outperforms both (60% success). At radius 0.035, Lifelong Adaptation is the only method to achieve non-zero success (20%). This supports our hypothesis that lifelong adaptation allows the agent to accumulate domain knowledge across episodes, fine-tuning the normalization statistics to the specific target goal configuration.

Lastly, beyond radius 0.04, all methods degrade to 0% success. However, Lifelong Adaptation maintains higher average rewards (221.1 vs 209.0 for Episodic) even in this failure mode, suggesting it remains closer to the solution.

C. Discussion on Adaptation Mechanism

The success of adaptation on goal shifts, where robot dynamics are invariant, suggests that by updating the LayerNorm statistics to match the test-time observation statistics, the encoder re-centers the OOD inputs into the range expected by the frozen policy. The superiority of Lifelong Adaptation implies that these statistics are stable across episodes for a fixed goal, and resetting them (Episodic) discards valuable calibration progress.

VI. Conclusion

In this work, we demonstrated that augmenting RL policies with a self-supervised ID auxiliary task serves a dual purpose: it acts as a powerful regularizer during training and enables effective adaptation during deployment.

First, we found that joint training alone significantly enhances robustness. The shared backbone learned via the auxiliary task proved sufficient to handle substantial variations in physical dynamics (e.g., link mass), rendering active adaptation unnecessary for those specific shifts. Second, for distributional shifts where the learned representation breaks down, such as severe goal displacement, our Lifelong TTA strategy successfully recovered performance. By updating only the normalization layers, the agent aligned the OOD observations with its training distribution.

While our results on the Reaching task establish the efficacy of this framework, future work should extend this analysis to contact-rich manipulation tasks, such as Pushing or Pick-and-Place. These environments involve complex frictional dynamics and object interactions that would likely reveal the full potential of dynamics-based adaptation, providing a more rigorous testbed for handling unseen physical environments.

REFERENCES

[1] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang, "Self-supervised policy adaptation during deployment," 2021. [Online]. Available: https://arxiv.org/abs/2007.04309

[2] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, and S. Bauer, "Causalworld: A robotic manipulation benchmark for causal structure and transfer learning," 2020. [Online]. Available: https://arxiv.org/abs/2010.04296

[3] X. Wang and Y. Jin, "Exploring causalworld: Enhancing robotic manipulation via knowledge transfer and curriculum learning," 2024. [Online]. Available: https://arxiv.org/abs/2403.17266

[4] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller, "Deepmind control suite," 2018. [Online]. Available: https://arxiv.org/abs/1801.00690

[5] M. Wüthrich, F. Widmaier, F. Grimminger, J. Akpo, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, J. Viereck, M. Naveau, L. Righetti, B. Schölkopf, and S. Bauer, "Trifinger: An open-source robot for learning dexterity," 2021. [Online]. Available: https://arxiv.org/abs/2008.03596