Dynamic Policy Anchoring To Prevent Catastrophic Forgetting In Reinforcement Learning

Krutik Patel
IIIT, Bangalore

Sai Madhavan G
IIIT, Bangalore

Tulika Saha
IIIT, Bangalore

Shrisha Rao
IIIT, Bangalore

***Abstract*—Catastrophic forgetting remains a significant challenge in reinforcement learning (RL), particularly in continual learning settings where agents must adapt to new tasks while preserving performance on previously learned ones. We propose *Dynamic Policy Anchoring* (DPA), a novel approach that extends the Proximal Policy Optimization (PPO) algorithm by incorporating an anchor policy to mitigate knowledge degradation across tasks. Our method evaluates candidate anchor policies based on task-specific performance metrics and adapts the PPO objective to constrain policy divergence during training. Although task change detection is critical to DPA, our experiments reveal challenges in reliably detecting transitions through both reward- and state-based methods.**

We evaluate our approach on a continual learning scenario involving modified versions of the HalfCheetah environment. While initial results demonstrate promise, with $\lambda = 1$ outperforming baselines in certain experiments, issues such as agent flipping and inconsistent performance hinder broader conclusions. Efforts to resolve flipping through reward shaping and episodic termination proved ineffective, highlighting the need for further refinement and experimentation. Despite its limitations, DPA offers a foundation for exploring catastrophic forgetting in RL and contributes valuable insights to the field.

I. Introduction

Transfer learning in reinforcement learning (RL) enables agents to reuse knowledge from previously solved tasks to accelerate learning on new tasks [1]. This capability is essential for developing efficient, scalable RL systems for real-world applications, where training from scratch for every task is often impractical. However, RL agents remain vulnerable to *catastrophic forgetting* [2], where learning new tasks disrupts prior knowledge, leading to significant performance degradation on earlier tasks. Addressing this issue is particularly challenging in continual learning scenarios, where an agent must navigate sequences of tasks with minimal external supervision.

In this work, we explore the potential of *Dynamic Policy Anchoring* (DPA), a technique designed to mitigate forgetting in sequential task learning. DPA builds on the *Proximal Policy Optimization* (PPO) algorithm [3], extending it with a mechanism to preserve knowledge from prior tasks. The proposed approach involves selecting a high-performing policy from the previous task as an anchor and constraining the deviation of the current policy from this anchor using a policy distance penalty term, measured via KL-Divergence. The goal is to retain knowledge from earlier tasks while adapting to new ones.

While our initial implementation focuses on environments where task boundaries are clearly defined, the ultimate vision of DPA includes task detection in online, unsupervised settings. Although we did not fully implement task detection in this study, we discuss the conceptual framework and challenges involved. Similarly, while we provide preliminary results, further empirical investigation is required to substantiate the effectiveness of DPA compared to existing approaches.

Through this work, we aim to contribute to the ongoing dialogue around overcoming catastrophic forgetting in RL, proposing a promising direction for future exploration and refinement.

II. Related Work

Catastrophic forgetting, first identified in neural networks by [2], describes the phenomenon where learning new tasks causes the network to lose proficiency on previously learned tasks. For example, a network trained on a multi-label dataset exhibited degraded performance on the original dataset after being fine-tuned on a new multi-label dataset [2], [6]. This challenge is especially pronounced in continual learning, where systems must learn sequentially without direct access to prior training data.

Learning without Forgetting (LwF) [6] introduced an innovative solution for classification tasks by utilizing task-specific output layers while sharing hidden layers across tasks. During training on new tasks, the approach uses soft targets from previous tasks as a regularization term to penalize deviations in the network's outputs, thereby retaining knowledge from earlier tasks.

In reinforcement learning (RL), catastrophic forgetting is defined as an agent's declining performance on previously mastered tasks after acquiring new experiences. This issue becomes more complex when new tasks are represented as distinct Markov Decision Processes (MDPs) with differing state and action spaces. For example, new tasks could range from entirely new environments to variations within the same environment, such as a new level in a video game.

Several methods have been proposed to address catastrophic forgetting in RL. PLAiD [7] employs progressive distillation, where a teacher network trained on older tasks transfers its knowledge to a student network learning a new task. This approach ensures that knowledge from previous tasks is incorporated into the training process for new tasks, mitigating forgetting.

Kaplanis et al. [4] extended this idea by introducing a loss function that minimizes the divergence between consecutive policies learned by an agent. By preserving the similarity between adjacent policies, their method ensures that the agent consolidates prior knowledge while adapting to new tasks. This approach effectively retains performance on previous tasks while facilitating continual adaptation, addressing the unique challenges of catastrophic forgetting in RL.

III. Problem Formulation

Consider a sequence of experiences $E = (e_1, e_2, \ldots, e_T)$, where each experience $e_t = \langle s_t, a_t, r_t, s_{t+1} \rangle$ represents a single interaction with the environment. This sequence spans n distinct tasks $(T_1, \ldots, T_n)$, where each task T_i is defined by a unique Markov Decision Process (MDP) $\langle \mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{R}_i, \gamma_i \rangle$. For simplicity, we assume shared state ($\mathcal{S}$) and action ($\mathcal{A}$) spaces across tasks, while allowing the transition dynamics ($\mathcal{P}_i$) and reward functions ($\mathcal{R}_i$) to vary between tasks.

This setting presents two core objectives:

1) **Task Boundary Identification:** Develop a function $f : E \to \{1, \ldots, n\}^T$ that maps each experience e_t to its corresponding task index. This function partitions E into n subsequences, where each subsequence corresponds to the experiences from a single task. Identifying task boundaries is critical in the absence of explicit signals, as it determines how the agent structures its learning.
2) **Sequential Policy Optimization with Retention:** Optimize a policy π_θ, parameterized by θ, sequentially across the identified tasks while preserving performance on previously learned tasks. Specifically, when learning task T_i, the agent must ensure:

$$J_j(\pi^j) - J_j(\pi_\theta) < \delta, \ \forall j < i$$

where $J_j(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma_j^t r_t \mid T_j \right]$ is the expected cumulative reward under task T_j, π^j is the optimal policy for task T_j, and $\delta > 0$ is a threshold governing the acceptable performance degradation.

A. Experimental Environment

Our approach is particularly effective in scenarios where sequential tasks share similar state-space dimensionality but exhibit significant differences in their optimal policy regions. This distinction is critical because excessive overlap in optimal state-action mappings between tasks exacerbates catastrophic forgetting. Neural network function approximators typically maintain a single mapping per state vector, which limits their ability to handle such overlaps effectively.

For our empirical evaluation, we adapt two related tasks from [4] using the MuJoCo physics engine [5]. The base environment is the HalfCheetah from OpenAI Gym, a simulated two-dimensional robot consisting of a torso and two articulated legs. The agent's state space comprises 17 dimensions, including joint angles, velocities, and center of mass information. Its action space consists of 6 continuous values controlling joint torques. The reward function encourages forward velocity while penalizing excessive control effort.

To define the second task, we modify the HalfCheetah environment by increasing the leg segment lengths by 25%, creating a variation we call *HalfCheetahBigLeg*. This alteration preserves the locomotion objective while significantly changing the optimal policy. For clarity, we refer to the original environment as *HalfCheetahVanilla*.

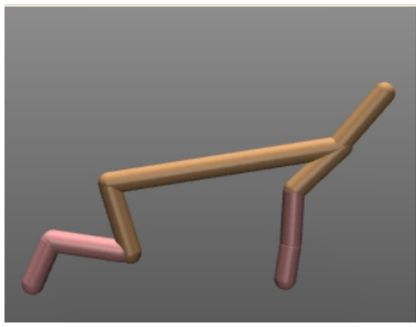


Fig. 1. Vanilla HalfCheetah Environment. This image shows the original HalfCheetah robot used in the experiments.

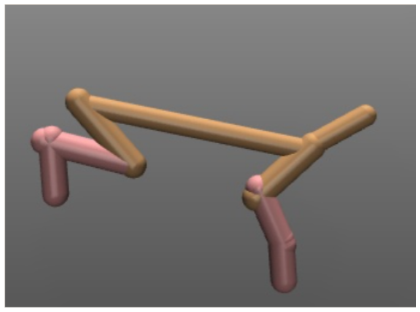


Fig. 2. Big Leg HalfCheetah Environment. This image illustrates the modified HalfCheetah robot with increased leg segment lengths.

IV. Dynamic Policy Anchoring

A. Policy Quality Assessment

To mitigate catastrophic forgetting, it is crucial to identify and retain high-quality policies from the current task as candidate anchor policies. We evaluate the quality of a policy π_θ for task T_i using the following criterion:

$$\text{IsGoodPolicy}(\pi_\theta, T_i) := \mathbb{E}[R(\pi_\theta, T_i)] \geq \tau_i$$

Here, $\mathbb{E}[R(\pi_\theta, T_i)]$ represents the expected cumulative reward obtained by policy π_θ on task T_i, and τ_i is a predefined threshold indicating the minimum acceptable performance for the task. Policies meeting this criterion are considered as potential anchor policies for preserving knowledge during subsequent task training.

B. Task Change Detection

Detecting task transitions is critical in continual reinforcement learning, as it allows the agent to appropriately adapt its learning process and preserve performance across tasks. Without explicit task boundary signals, task change detection becomes a challenging yet essential component for mitigating catastrophic forgetting.

We explored two approaches to address task change detection:

1) Rewards-Based Detection:: In this method, we monitored the accumulated rewards per episode. A consistent decline in rewards was flagged as an indicator of a potential task transition. However, this approach was unreliable due to the inherent instability of the environment, which led to frequent false positives. This instability made it difficult to distinguish genuine task changes from noise in the reward signal.

2) State-Based Detection:: Our second approach modeled the state space using a Gaussian Mixture Model (GMM). By calculating the Mahalanobis distance for each observed state, we attempted to identify out-of-distribution (OOD) states, which we treated as indicators of a task transition. Unfortunately, this method also proved ineffective. The modeling process involved numerous hyperparameters, and the resulting models failed to provide reliable detection, further complicating the task change detection process.

Both methods highlighted the complexity of detecting task transitions in dynamic environments, emphasizing the need for more robust and adaptive detection techniques.

C. Dynamic Policy Anchoring

The core idea behind Dynamic Policy Anchoring (DPA) is to adapt the Proximal Policy Optimization (PPO) objective function by penalizing the distance between the current policy and a selected anchor policy from previous tasks. The goal is to retain performance on previously learned tasks while adapting to new ones. This is achieved by incorporating a penalty term based on the Kullback-Leibler (KL) divergence between the current policy and the anchor policy.

The objective function for DPA is:

$$L_{\text{ANCHOR}}(\theta) = L_{\text{CLIP}}(\theta) + \lambda(t) \cdot D_{\text{KL}}(\pi_\theta, \pi_{\text{anchor}})$$

Here, $L_{\text{CLIP}}(\theta)$ represents the standard PPO objective, and $\lambda(t)$ is a time-dependent weight that adjusts the influence of the anchor penalty over time. The term $D_{\text{KL}}(\pi_\theta, \pi_{\text{anchor}})$ represents the KL divergence between the current policy π_θ and the anchor policy π_{anchor}, which penalizes large deviations between the two policies.

The KL divergence is computed as:

$$D_{\text{KL}}(\pi_\theta, \pi_{\text{anchor}}) = \mathbb{E}_{s\sim\rho}\left[\text{KL}(\pi_{\text{anchor}}(\cdot|s) \parallel \pi_\theta(\cdot|s))\right]$$

where ρ is the distribution over states, and the KL term measures the difference in policy distributions over the state space. By minimizing this divergence, the agent preserves the learned behavior of previous tasks while adapting to new ones.

The PPO clip objective is defined as:

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

where $r_t(\theta)$ is the probability ratio of the current policy to the previous policy, and $\hat{A}_t$ is the estimated advantage at time step t. The clip operation ensures that the objective function remains within a trust region, preventing excessively large policy updates.

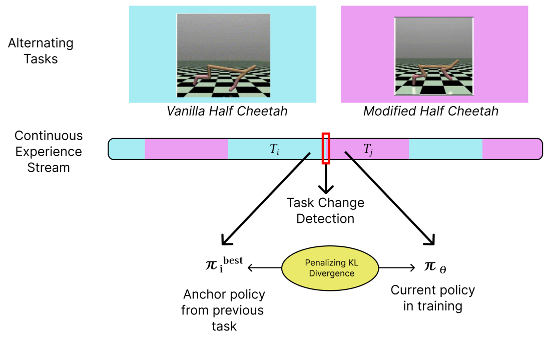


Fig. 3. Architecture Diagram of Dynamic Policy Anchoring (DPA). This diagram illustrates the integration of the anchor policy with the PPO framework.

V. Experiments

We evaluate the proposed Dynamic Policy Anchoring (DPA) approach on a continual learning scenario using the HalfCheetah environment. Following the setup in [4], training alternates between two tasks, *HalfCheetahVanilla* and *HalfCheetahBigLeg*, every 1 million timesteps, for a total of 20 million timesteps. Due to the challenges faced with task change detection, as described earlier, we hardcoded the task change points into the training loop. This ensured that the agent switched to the appropriate task without relying on unreliable detection mechanisms.

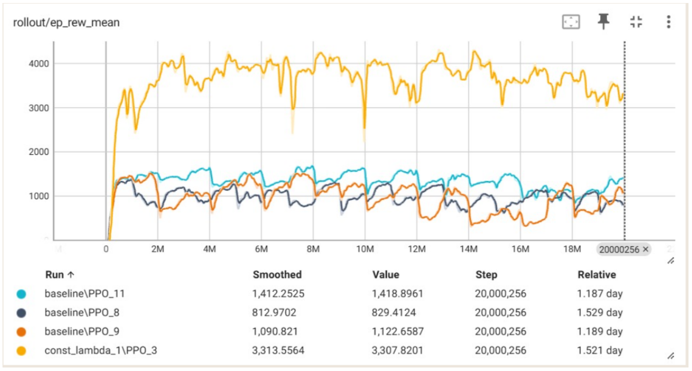
For evaluation, we tracked the reward per episode over the course of training and compared our method against three baseline models. We then experimented with different constant values of λ in the modified PPO objective to assess the impact of the anchor policy constraint. The results, visualized in the plot of reward per episode over 20 million timesteps, indicate that the yellow line corresponding to $\lambda = 1$ outperforms the baseline models (blue, orange, and indigo). Notably, the agent's performance did not degrade when the environment changed, demonstrating the promise of our approach in retaining knowledge across tasks.

However, repeated experiments with the same hyperparameters yielded inconsistent results. We observed that during training, the agent occasionally exhibited a flipping behavior, where its head hit the ground, leading to drastically reduced rewards. This flipping phenomenon was prevalent across both baseline and modified models, complicating the interpretation of our findings.

A. Addressing Flipping

To address the flipping issue, we experimented with several strategies:

1) Assigning a large negative reward when the agent's head touched the ground.



	Run ↑	Smoothed	Value	Step	Relative
●	baseline\PPO_11	1,412.2525	1,418.8961	20,000,256	1.187 day
●	baseline\PPO_8	812.9702	829.4124	20,000,256	1.529 day
●	baseline\PPO_9	1,090.821	1,122.6587	20,000,256	1.189 day
●	const_lambda_1\PPO_3	3,313.5564	3,307.8201	20,000,256	1.521 day

Fig. 4. Reward per episode over 20 million timesteps. The yellow line, corresponding to $\lambda = 1$, performs better than the baseline models (blue, orange, and indigo).

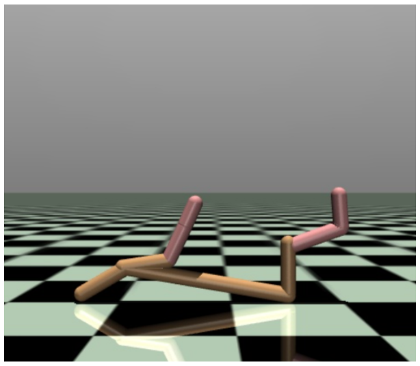


Fig. 5. Visualization of the flipping behavior. The agent's head hits the ground, resulting in a significant drop in performance.

2) Terminating the episode immediately if the agent flipped.
3) Providing a positive reward proportional to the height of the agent's head.
4) Warm-starting the agent from a checkpoint before the flipping behavior emerged.

Unfortunately, none of these methods successfully resolved the flipping issue, and the agent continued to exhibit inconsistent behavior.

B. Discussion of Results

While initial results with $\lambda = 1$ were encouraging, the inability to reliably resolve flipping and the reliance on hard-coded task change points highlight significant limitations in our approach. These findings underscore the need for more robust methods to handle catastrophic forgetting and emergent behaviors in reinforcement learning. Further experimentation is required to validate the effectiveness of DPA in more complex and diverse environments.

VI. Conclusion

In this work, we addressed the problem of catastrophic forgetting in reinforcement learning through the introduction of *Dynamic Policy Anchoring* (DPA), an extension of the Proximal Policy Optimization (PPO) algorithm. By leveraging anchor policies to constrain policy updates, our approach aimed to preserve knowledge from previously learned tasks while enabling adaptation to new ones.

Key components of our method include a policy quality assessment mechanism for identifying suitable anchor policies and an adaptation of the PPO objective to penalize policy divergence. Despite its conceptual soundness, implementing DPA faced challenges, particularly in task change detection. Neither reward-based nor state-based methods for detecting task boundaries proved reliable, with each approach yielding false positives or failing under complex environmental dynamics.

Empirical evaluations on continual learning scenarios using the HalfCheetah environment provided mixed results. While $\lambda = 1$ demonstrated better performance compared to baselines in certain experiments, issues such as agent flipping and inconsistent replicability of results revealed significant limitations. Attempts to mitigate flipping through various interventions, including reward shaping and episodic termination, were unsuccessful, further complicating the validation of our approach.

In conclusion, while DPA shows promise as a potential solution for mitigating catastrophic forgetting, our findings highlight the complexity of achieving reliable continual learning in reinforcement learning. Addressing task change detection, improving stability during training, and resolving emergent behaviors like flipping remain critical areas for future research. This work lays a foundation for exploring more robust and scalable solutions to catastrophic forgetting in RL.

References

[1] Z. Zhu, K. Lin, A. K. Jain and J. Zhou, ”Transfer Learning in Deep Reinforcement Learning: A Survey,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 11, pp. 13344-13362, 1 Nov. 2023, doi: 10.1109/TPAMI.2023.3292075.

[2] McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pp. 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8.

[3] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[4] Kaplanis, C., Shanahan, M., and Clopath, C. (2019). Policy consolidation for continual reinforcement learning. ICML 2019, pp. 3242–3251. http://proceedings.mlr.press/v97/kaplanis19a.html.

[5] E. Todorov, T. Erez and Y. Tassa, ”MuJoCo: A physics engine for model-based control,” 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 5026-5033, doi: 10.1109/IROS.2012.6386109.

[6] Z. Li and D. Hoiem, “Learning without Forgetting“, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 12, pp. 2935-2947, 1 Dec. 2018, doi: 10.1109/TPAMI.2017.2773081

[7] Berseth, G., Xie, C., Cernek, P., and de Panne, M. V. (2018). Progressive reinforcement learning with distillation for multi-skilled motion control. ICLR 2018. https://people.eecs.berkeley.edu/ gberseth/projects/PLAiD/paper.pdf.