Approximate Next Policy Sampling: Replacing Conservative Target Policy Updates in Deep RL

Dillon Sandhu, Ronald Parr

Keywords: Foundations, Policy Improvement

Summary

Conservative policy updates (small changes in policy space) are an integral part of most modern RL algorithms. The origins trace back to a classic "chicken-and-egg" problem in policy improvement: to safely improve a policy, the value function must be accurate on average over the distribution of states encountered by policy *after the update*. To deal with this, conservative methods constrain the policy to remain where the value function estimate is already assumed to be trustworthy.

This paper explores an alternative, *Approximate Next Policy Sampling* (ANPS). By (approximately) sampling the states that the policy will encounter after the update, ANPS can train the value function on this importance set of states. We implement ANPS with *Stable Value Approximate Policy Iteration* (SV-API), a lightweight modification to standard RL that holds the target policy fixed for multiple rounds of data collection, only committing to a policy update once the value estimates have stabilized. SV-API matches or exceeds baseline performance on high-dimensional discrete (Atari) and continuous control, while making substantially larger target policy updates. These results demonstrate the viability of ANPS as an alternative approach to a classic challenge in RL.

Contribution(s)

1. Proposes Approximate Next Policy Sampling: an alternative to conservative policy updates that explicitly aligns the training distribution with the next policy's state visitation distribution – rather than constraining the policy update.
   Context: As implemented, ANPS allows for unconstrained target policy updates, but must constrain the behavioral policy updates.
2. A general bound (Theorem 3.3) that isolates the importance of the next policy's distribution. This shows why conservative updates are only one possible solution to the distribution mismatch problem raised by Kakade & Langford (2002).
   Context: This bound follows straightforwardly from the Performance Difference Lemma Kakade & Langford (2002).
3. SV-API, a practical modification to standard online RL that approximately samples the next policy by decoupling the behavioral and target policies, along with empirical evidence that this allows PPO to execute larger policy jumps without increasing the risk of policy degradation due to distribution shift.
   Context: SV-API is an proof-of-concept implementation of ANPS. It relies on a proxy metric (stability of the value estimates) to determine when the value function is accurate enough to update the target policy. Furthermore, it requires off-policy evaluation since it introduces a separate behavioral policy.
4. A bound (Theorem 4.1) proving that our SV-API guarantees policy improvement by controlling training error and behavioral divergence.
   Context: Like standard related bounds (Kakade & Langford (2002), Schulman et al. (2015)), it assumes bounded value error, which is not guaranteed for deep RL.

Approximate Next Policy Sampling: Replacing Conservative Target Policy Updates in Deep RL

Dillon Sandhu[1], **Ronald Parr** [1]
dillon.sandhu@duke.edu

[1]**Department of Computer Science, Duke University, USA**

Abstract

We revisit a fundamental "chicken-and-egg" problem in reinforcement learning: policy improvement depends on the state visitation distribution of the updated policy, which is unknown at training time. Conservative updates solve this problem, but at the cost of shrinking the policy update. This paper explores an alternative solution, Approximate Next Policy Sampling (ANPS), which modifies the training distribution to approximately equal the next policy's state visitation distribution. To demonstrate the feasibility and efficacy of ANPS, we introduce Stable Value Approximate Policy Iteration (SV-API), which uses an iteratively updated behavioral policy to gather data, only committing to a new policy once a convergence condition has been met. The update is guaranteed to be safe if certain stability criteria are met, otherwise, it is no less safe than standard approximate policy iteration. We find the application of SV-API to PPO matches or improves performance on high-dimensional control (Atari) and continuous control benchmarks. Moreover, it executes substantially larger target policy updates. These results demonstrate the viability of ANPS as a new solution to a classic problem.

1 Introduction

Policy Iteration (PI; Howard (1960)) has long been the "big hammer" for solving exact MDPs, often returning the optimal policy after surprisingly few iterations. In each round, PI computes the action-value function of a fixed policy, Q^π. It then obtains the *next policy* (π') via the greedy policy update – selecting an action to maximize Q^π at each state. **Approximate Policy Iteration** (API) replaces exact computation of Q^π with approximation (Bertsekas & Tsitsiklis, 1996). However, this undermines the fundamental reason PI works: monotonic policy improvement (Bertsekas & Tsitsiklis (1996), Perkins & Precup (2002)). Small changes to the value function estimate can lead to massive shifts in the state-visitation distribution (Perkins & Precup (2002)). Unless the estimated action-value is accurate on the new state distribution, improvement cannot be guaranteed, and API may fail to converge to a locally optimum policy.

Ensuring policy improvement for API is a classic chicken-and-egg problem. Performance decline can occur even with a "well-behaved" function approximator that minimizes error on its training set—indeed, this behavior is part of the problem. By fitting the current distribution, the approximator may have high error on rare states. Since these states may be frequented after the update, improvement cannot be guaranteed. Perhaps counterintuitively, the ideal training distribution is the state-visitation distribution of the *next* policy, which is only defined after the update: hence the "chicken-and-egg" analogy.

Various mechanisms for enforcing small policy changes have become widely adopted means of mitigating this problem. **Conservative policy updates** (Kakade & Langford (2002), Schulman et al.

(2015)) make restricted steps in policy space, preventing drastic shifts in the state-visitation distribution. This strategy is foundational to many modern actor-critic algorithms (Schulman et al. (2015), Schulman et al. (2017), Abdolmaleki et al. (2018), Hessel et al. (2022)). However, in minimizing the worst case policy degradation, these approaches also limit the potential upside of performance improvement.

Ultimately, the problem is not the size of the policy update, but the value error on the next policy's state distribution. This paper adopts an approach based on sampling, which we call **Approximate Next Policy Sampling** (ANPS). Rather than forcing the next policy π' to remain close to π, ANPS draws training data from a distribution designed to cover the distribution of states that π' visits. This focuses value estimation on the states that are most important for performance.

We implement ANPS with *Stable Value Approximate Policy Iteration* (SV-API), a variant of API that uses multiple rounds of sampling and fitting during policy evaluation. Holding the target policy fixed, SV-API repeatedly collects trajectories from a behavioral policy, which is updated based on the present value estimate. When value estimate on the resulting state distribution converges, or a patience threshold is met, the newest behavioral policy is selected to be π'. Empirically, we demonstrate that SV-API applied to PPO (SV-PPO) improves performance on standard benchmarks Schulman et al. (2017). Analysis of our results reveals that SV-PPO shrinks the change in behavioral policy, while executing substantially larger target policy updates. This decoupling of the target policy from the behavioral policy reveals that small policy updates are useful primarily for safely exploring the state space, while changes to the target policy are not essential, and can even be detrimental.

2 Background

We consider a Markov Decision Process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$. A policy $\pi(\cdot|s)$ is a distribution over actions given state. At each timestep t, the agent takes action $a_t \sim \pi(\cdot|s_t)$, transitioning to next state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and receives bounded reward $r_t = r(s_t, a_t)$ with $r_t \in [0, 1]$. We assume discount factor $\gamma \in [0, 1)$, and fixed initial state s_0. We overload notation and sometimes use $\pi(s)$ as shorthand for $\pi(\cdot|s)$.

The RL problem is to find a policy that maximizes the expected performance from the start state, which is given by the **value function** $V^\pi(s_0) \doteq \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) \mid s_0\right]$.[1] The **action-value function**, Q^π, and **advantage function** are related value functions, defined as follows:

$$Q^\pi(s, a) \doteq r(s, a) + \gamma \mathbb{E}_{s' \sim P(s,a)} V^\pi(s) \tag{1}$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} Q^\pi(s, a) \tag{2}$$

$$A^\pi(s, a) \doteq Q^\pi(s, a) - V^\pi(s). \tag{3}$$

Since the reward is bounded, so is the value function: $\max_{\pi,s,a} Q^\pi(s, a) \leq \sum_{i=0}^\infty \gamma^i = \frac{1}{1-\gamma}$.

A convenient way to express a policy's performance is in terms of its discounted **state action visitation distribution**. This quantity discounts states based on how far they are in the future, and is normalized by $(1 - \gamma)$ to ensure it sums to one.

$$d^\pi(s, a) \doteq (1 - \gamma) \sum_{t=0}^\infty \gamma^t P(s_t = s, a_t = a)$$

The Performance Difference Lemma (PDL; Kakade & Langford (2002)) writes the improvement of arbitrary policy π' over arbitrary policy π as an expected advantage, with respect to $d^{\pi'}$.

$$V^{\pi'}(s_0) - V^\pi(s_0) = \frac{1}{1 - \gamma} \mathbb{E}_{s,a \sim d^{\pi'}} A^\pi(s, a). \tag{PDL}$$

Finally, for any two distributions p and q over random variable x, the **total variation distance** is defined as $d_{TV} \doteq \frac{1}{2} \sum_x |p(x) - q(x)|$.

[1] The subscript π indicates the expectation is over probabilities that are jointly governed by π and P.

3 Distribution shift in on-policy API

Most scalable RL algorithms update the policy to maximize performance, as measured by the value estimate. Algorithm 1 shows this high-level procedure, On-Policy API, which constructs a sequence of policies $\{\pi_k\}_{k=0}^{\infty}$ by iterating between data collection due to π_k, policy evaluation of π_k, and updating the policy to obtain π_{k+1}. The term **on-policy** means that the dataset $\mathcal{D}_k$ contains samples due to the **target policy** π_k.

We assume generic subroutines for value function estimation and policy update: `Eval` estimates $q^{\pi_k} \approx Q^{\pi_k}$, while Γ returns a new policy. Since it estimates the value of the target policy from data, `Eval` is a function of the dataset $\mathcal{D}_k$ as well as target policy π_k. The policy improvement operator, Γ, is a function of q^{π_k}, plus optional arguments π_k and $\mathcal{D}_k$. With these methods broadly defined, Algorithm 1 encompasses most on-policy actor-critic and value-based RL.

Algorithm 1 On-Policy API

Require: API Subroutines `Eval` and Γ

Initialize π_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect dataset $\mathcal{D}_k$ by rolling out π_k.
  $q_k^{\pi_k} \leftarrow \texttt{Eval}(\pi_k, \mathcal{D}_k)$
  $\pi_{k+1} \leftarrow \Gamma(q_k^{\pi_k}, \pi_k, \mathcal{D}_k)$
end for

Algorithm 2 Stable Value API

Require: `Eval`, Γ, and Stability Criterion $\mathcal{C}$

Initialize π_0, $\beta_0 \leftarrow \pi_0$
for $k = 0, 1, 2, \ldots$ **do**
  Collect dataset $\mathcal{D}_k$ by rolling out β_k
  $q_k^{\pi_k} \leftarrow \texttt{Eval}(\pi_k, \mathcal{D}_k)$
  $\beta_{k+1} \leftarrow \Gamma(q^{\pi_k}, \beta_k, \mathcal{D}_k)$
  if $\mathcal{C}(q_k^{\pi_k}, q_{k-1}^{\pi_{k-1}}, \mathcal{D}_k)$ is **True then**
    $\pi_{k+1} \leftarrow \beta_{k+1}$ ▷ Update target policy
  else
    $\pi_{k+1} \leftarrow \pi_k$ ▷ Hold target policy
  end if
end for

Figure 1: API (Left) and Stable Value API (Right).

We now formalize the distribution shift problem by quantifying the improvement due to a single iteration of Algorithm 1. The bound obtained in this section reveals the importance of aligning the training distribution and next-policy's distribution, a concept we term *Next Policy Alignment*. We begin by defining two quantities upon which the policy improvement at each round of Algorithm 1 depends.

Definition 3.1 (Action-Value Error)**.** The μ-weighted estimation error of q^π is

$$\varepsilon(\mu, q^\pi) \doteq \sum_{s \in S} \sum_{a \in A} \mu(s,a) \left| Q^\pi(s,a) - q^\pi(s,a) \right|. \tag{4}$$

Definition 3.2 (Estimated Expected Advantage)**.** The estimated expected advantage of arbitrary π' over target policy π is

$$\mathbb{A}_{\pi'}^{\pi} \doteq \mathbb{E}_{s,a \sim d^{\pi'}} \left[q^\pi(s,a) - V^\pi(s) \right]. \tag{5}$$

Most policy improvement operators optimize a quantity like Equation 5. For example, if the policy update is greedy with respect to q^π, then π' maximizes the quantity in Definition 3.2 at all states exactly. We now bound the policy improvement of π' relative to π.

Theorem 3.3 (Value Error and Policy Improvement)**.** *Given action-value estimate q^π and target policy π, let the next policy be $\pi' = \Gamma(q, \pi)$. Then,*

$$\frac{1}{1-\gamma}\left(\mathbb{A}_{\pi'}^{\pi} - \varepsilon(d^{\pi'}, q^\pi)\right) \leq V^{\pi'}(s_0) - V^\pi(s_0) \leq \frac{1}{1-\gamma}\left(\mathbb{A}_{\pi'}^{\pi} + \varepsilon(d^{\pi'}, q^\pi)\right) \tag{6}$$

This theorem implies that $\mathbb{A}^{\pi}_{\pi'}$ must be larger than $\varepsilon(d^{\pi'}, q^{\pi})$ to guarantee monotonic policy improvement. The proof is in Appendix (6). It starts from the Performance Difference Lemma (PDL), and replaces $A^{\pi}(s,a)$ with an interval of radius $\varepsilon(d^{\pi'}, q^{\pi})$ centered at $q(s,a) - V(s)$.

We assume that `Eval` minimizes the action-value error on the training distribution, μ. Thus, $\varepsilon(d^{\pi'}, q^{\pi})$ must be controlled some other way. The following Lemma shows that when the training distribution matches the next round's on-policy distribution, then $\varepsilon(d^{\pi'}, q^{\pi})$ can be controlled with standard distribution-shift arguments.

Lemma 3.4. *Assume bounded value error:* $\max_{s,a,\pi} |Q^{\pi}(s,a) - q^{\pi}(s,a)| \leq \frac{1}{1-\gamma}$. *For any two state-action distributions* μ *and* $d^{\pi'}$*:*

$$\varepsilon(d^{\pi'}, q^{\pi}) \leq \varepsilon(\mu, q^{\pi}) + \frac{2}{1-\gamma} d_{TV}(d^{\pi'}, \mu)$$

The proof is in Appendix A. Combined with the preceding Theorem 3.3, Lemma 3.4 implies that that, if the distribution shift is limited, then the policy improvement can be lower bounded. We call the degree of distribution shift between training distribution and policy improvement Next Policy Alignment.

Definition 3.5 (Next Policy Alignment (NPA)). Given next policy π', a training distribution $\mu(s,a)$ satisfies δ_{dist}-NPA if $d_{TV}(d^{\pi'}, \mu) \leq \delta_{dist}$.

Corollary 3.6. *Assume the conditions of Theorem 3.3 and Lemma 3.4 hold, and that training distribution* μ *achieves* δ_{dist}*-NPA (*$d_{TV}(d^{\pi'}, \mu) \leq \delta_{dist}$*). Then, the policy improvement is at least:*

$$V^{\pi'}(s_0) - V^{\pi}(s_0) \geq \frac{1}{1-\gamma} \mathbb{A}^{\pi}_{\pi'} - \frac{1}{1-\gamma} \varepsilon(\mu, q^{\pi}) - \frac{2}{(1-\gamma)^2} \delta_{dist} \tag{7}$$

Proof. Starting from Theorem 3.3, upper bound $\varepsilon(d^{\pi'}, q^{\pi})$ using Lemma 3.4. □

When δ_{dist}-NPA is satisfied, then the policy improvement suffers two penalties: one proportional to δ_{dist}, and the other proportional to the training data error. The relationship with δ_{dist} in Cor. 3.6 is the same in bounds due to Kakade & Langford (2002), which were later adopted by Schulman et al. (2015). That result and our result both share a $(1-\gamma)^{-2}$ coefficient in front of the distribution shift term. Unlike that work, our bound explicitly quantifies the dependence on value error.

3.1 Conservative Policy Iteration (CPI)

Conservative policy updates are designed to ensure that the policy improvement operator in Algorithm 1 satisfies δ-NPA by making small policy changes. The following Lemma due to Agarwal et al. (2021) relates the state-action distribution shift to the maximum policy change.

Lemma 3.7. $d_{TV}(\pi'(\cdot|s), \pi(\cdot|s)) \leq \delta(1-\gamma)$ *for all states* s *implies* $d_{TV}(d^{\pi'}(s,a), d^{\pi}(s,a)) \leq \delta$.

In the original CPI algorithm (Kakade & Langford, 2002), the next policy π' is a mixture of π and the greedy policy. Specifically, Γ_{CPI} sets $\pi_{k+1} = (1-\alpha)\pi_k + \alpha\pi_{greedy}$, limiting the policy divergence such that $d_{TV}(\pi'(\cdot|s), \pi(\cdot|s)) \leq \alpha$ at every state. The update $\pi' = \Gamma_{CPI}(q^{\pi}, \pi, D \sim d^{\pi})$ enjoys the following policy improvement bound due to Lemma 3.7 and Cor. 3.6.

$$V^{\pi'}(s_0) - V^{\pi}(s_0) \geq \frac{1}{1-\gamma} \mathbb{A}^{\pi}_{\pi'} - \frac{1}{1-\gamma} \varepsilon(d^{\pi}, q^{\pi}) - \frac{2\alpha}{(1-\gamma)^3} \tag{8}$$

From 8, assuming $\varepsilon(\mu, q^{\pi})$ is small enough, CPI can guarantee monotonic policy improvement, but at the cost of exceedingly small policy improvements. Because bounding distribution shift using policy divergence incurs a penalty of $\alpha/(1-\gamma)^3$, the permitted step size vanishes rapidly as $\gamma \to 1$. Assuming $\gamma = 0.99$, Theorem 12.2 of Agarwal et al. (2021) gives $\alpha \approx 0.0025$. According to Theorem 3.3, enforcing this limit wastes data and compute if the value error is low enough on the

next policy's state action distribution. Indeed, typical deep actor-critic methods implement much weaker conservatism than CPI theory suggests. For example, the Muesli (Hessel et al., 2022) update is regularized to prefer policies with $d_{TV}(\pi_k, \pi_{k+1}) \leq 0.46$, a jump in policy space large enough to make the policy improvement bound vacuous. As another example, Engstrom et al. (2020) found that TRPO and PPO held the average KL divergence at around 0.05 – still at least $20\times$ too large to guarantee policy improvement. While these methods are widely applied, they abandon strict theoretical safety (Stiennon et al. (2020), OpenAI (2023)). In the next section, we introduce a method that attempts to minimize $\varepsilon(d^{\pi'}, q^{\pi})$ directly.

4 Approximate Next Policy Sampling with Stable Value API

Approximate Next Policy Sampling (ANPS) achieves NPA through a fundamentally different mechanism: constructing a training distribution μ_k that explicitly approximates the next policy's state distribution, $d^{\pi_{k+1}}$. ANPS achieves this by selecting a behavioral policy whose state-action visitation distribution satisfies δ_{dist}-NPA. Then, minimizing the action-value error on the training data, $\varepsilon(\mu_k, q^{\pi_k})$, safely bounds the error on the crucial state-action visitation distribution of the next policy. This alleviates the need for π_{k+1} and π_k to remain close.

We introduce Stable Value API (SV-API, Algorithm 2) as a general recipe to execute ANPS with arbitrary Γ and `Eval`. SV-API separates the optimization process into three distinct policies at each iteration k:

- **Target Policy (π_k):** Uniquely determines the value target Q^{π_k} for evaluation.
- **Behavioral Policy (β_k):** Collects the data.
- **Next Target Policy (π_{k+1}):** Updated only after a convergence criterion is met.

The full algorithm is shown in Algorithm 2. In each round, the value estimate is trained on data from the behavioral policy, which is then updated with Γ. Unlike Algorithm 1, the *target policy* is updated only after a stability condition, $\mathcal{C}$, is met. In this work, we explore two concrete stability conditions:

1. **Static Intervals:** The target policy is updated every K rounds, where K is a hyperparameter.
2. **Dynamic Thresholding:** The change in the value estimates falls below a hyperparameter δ_v for at least K_{min} rounds. We track the absolute critic change:

$$\text{diff}_k \doteq \frac{1}{|\mathcal{D}|} \sum_{s,a \in \mathcal{D}} |q_k^{\pi_k}(s,a) - q_{k-1}^{\pi_{k-1}}(s,a)| \tag{9}$$

   This quantity is a measurable proxy for $\varepsilon(\mu_k, q_k^{\pi_k})$ and $d_{TV}(\mu_{k-1}, \mu_k)$. When either of these is large, diff_k may also be large. To help ensure transferability across environments, we normalize diff_k, dividing it by the return before comparing to δ_v. Finally, we implement a maximum number of rounds spent evaluating a fixed policy, K_{max}, to prevent stalling. The full convergence logic used in this work is shown in Algorithm 4,

Static interval thresholding can be obtained by setting the maximum and minimum number of iterations to the same value: i.e. $K_{min} = K_{max} = K$. We now present an experiment comparing the quantities from Section 2, for SV-API before returning to our theoretical analysis.

4.1 An empirical demonstration on the Four Rooms MDP

To examine how $\varepsilon(\mu_k, q^{\pi_k})$ and next-policy alignment interact in practice, we run deep RL on the Four Rooms environment (Sutton et al., 1999). We compare a conservative API method, PPO, and a Stable Value variant (SV-PPO), which performs Algorithm 2 with `Eval` and Γ from PPO. We defer the implementation details to Section 5. The start state is in the top left room, while the goal is the bottom right corner. The intended action is followed 80% of the time, otherwise the agent moves randomly. The layout of the environment is provided in the first panel of Figure 2, along with

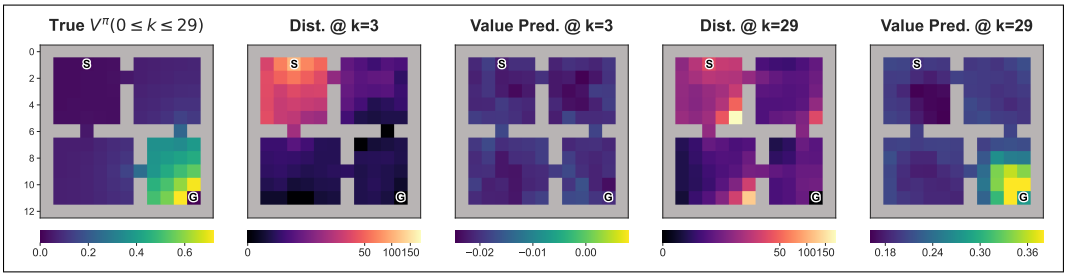


Figure 2: Snapshots of two rounds of SV-PPO on Four Rooms ($k = 3$ and $k = 29$) with a fixed target policy. True V^{π} indicates the true value function of the target policy. "Dist. @ k" is the moving average empirical state visitation distribution at round k. "Value Pred." is the predicted value.

the value function of the uniform random policy. Code to reproduce all experiments is available at https://github.com/dillonmsandhu/next-policy-sampling/.

Based on its convergence logic, SV-PPO delays updating the target policy until round $k = 30$. Figure 2 shows the true state value function V^{π_k} for $k \le 29$, as well as the state distribution of an early behavioral policy. Note that there is little overlap between the states visited with random exploration and states with high value under the random policy. Sampling with the behavioral policy remedies this. As the last two panels of Figure 2 show, by round $k = 29$, β_k has achieved significantly better coverage near the goal, leading to a much stronger value estimate compared to the earlier rounds. Without the sustained data collection from β_k, accurate value estimation of V^{π_0} at these distant, high-value states would be impossible.

In contrast, standard PPO immediately updates the policy based on initial, high-error value estimates. This leads to value churn (Tang & Berseth, 2024) and catastrophic forgetting around iteration 80. Figure 3a shows the corresponding performance collapse. A deeper analysis reveals the dynamic: the critic overestimates the value of rarely visited regions (the bottom room and leftmost wall) leading to degradation of the policy in these regions. Then, premature policy updates adjust the policy to avoid these regions, meaning that the value network rarely sees the data it would need to self-correct. Since the policy is weak in this region, but cannot avoid visiting it due to action stochasticity, PPO does not settle at the optimal policy during learning – as shown in the performance curve in Figure 3a. See Supplemental Materials F for a detailed analysis of PPO's and SV-PPO's learning dynamics.

By evaluating target policies for longer, SV-API suppresses value overestimation, obtains coverage of high-value states, and successfully converges to the optimal policy. Figure 3a plots the value of the sampling policy, while Figure 3b shows the value error, weighted by μ_{k+1}, the key quantity that can undermine policy improvement. Despite having less frequent target policy updates, both methods achieve a similar degree of NPA, as shown in Figure 3c. We track the convergence metric for SV-PPO in Figure 3d, observing that it's reasonably correlated with value error on the next behavioral policy's training distribution, the quantity for which it is a proxy.

4.2 Analysis of SV-API

We now consider when SV-API achieves a monotonic policy improvement. Since the target policy is only updated when a convergence criterion has been met, SV-API can improve it by ensuring the training distribution μ_k is similar to $d^{\pi_{k+1}}$ and lowering the training error. In other words, during update round k, the cumulative distribution of states visited by behavioral policies $\beta_{k' \le k}$ must provide good coverage of $d^{\pi_{k+1}}$, which, in an update round, equals $d^{\beta_{k+1}}$.

To simplify the analysis, we do not consider the entire training distribution, which was collected by $\beta_{k' \le k}$, but only the final d^{β_k}. With this simplification, a sufficient condition for monotonic policy

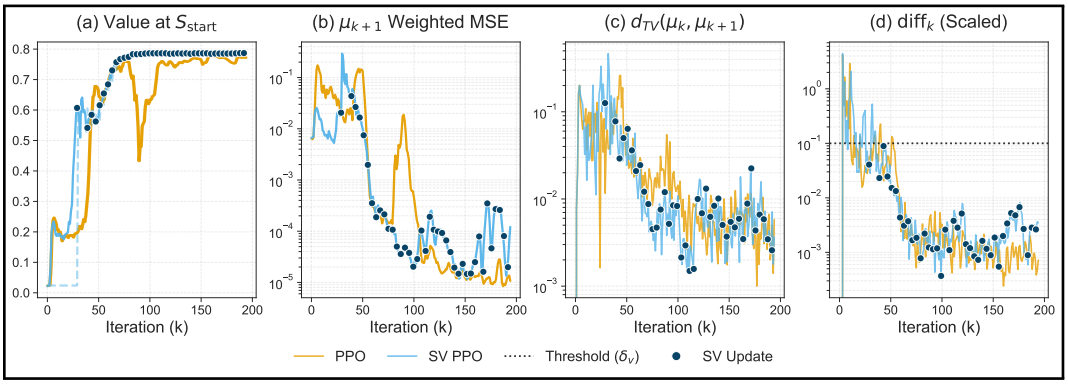


Figure 3: Four Rooms learning curves. Dots indicate target policy updates for SV-API. (a) Performance of the data-collection policy for both algorithms; (b) Weighted Mean Squared Value Error, weighted by the next policy's on-policy distribution, $\mathbb{E}_{\mu_{k+1}}(V^{\pi}(s) - v_k(s))^2$; (c) Total variation distance between successive training distributions μ_k; and (d) Convergence: scaled diff$_k$ vs. the threshold, δ_v (dashed line).

improvement is the proximity of *subsequent behavioral policies* (along with low value error). The theoretical mechanism for ANPS is iterative refinement of the behavioral policy until β_k and β_{k+1} are sufficiently similar, at which point $\mathcal{C}$ must trigger an update to the target policy. We now present our main theoretical result: sufficient conditions for SV-API to make safe, unconstrained updates to the target policy.

Theorem 4.1 (SV-API Improvement)**.** *In Algorithm 2, assume the target policy update occurs ($\pi_{k+1} \leftarrow \beta_{k+1}$), and that the behavioral policy has stabilized such that $\max_s d_{TV}(\beta_k(\cdot|s), \beta_{k+1}(\cdot|s)) \leq \delta(1-\gamma)$. Let the training data distribution μ_k be the stationary distribution induced by the behavioral policy, $\mu_k \doteq d^{\beta_k}$. Finally, assume bounded error on the training distribution $\varepsilon(\mu_k, q^{\pi_k}) \leq \varepsilon$ and globally $\max_{s,a,\pi} |Q^{\pi}(s,a) - q^{\pi}(s,a)| \leq \frac{1}{1-\gamma}$. Then, the performance improvement is at least:*

$$V^{\pi_{k+1}}(s_0) - V^{\pi_k}(s_0) \geq \frac{1}{1-\gamma}\left(\mathbb{A}^{\pi_k}_{\pi_{k+1}} - \varepsilon - \frac{2\delta}{1-\gamma}\right) \tag{10}$$

Proof. In an update round, the target policy becomes the new behavioral policy: $\pi_{k+1} = \beta_{k+1}$. The divergence between $d^{\beta_{k+1}}$ and μ_k is at most δ (Lemma 3.7). Plugging this in for δ_{dist} in Corollary 3.6 yields the result. □

It is instructive to contrast the bound in Theorem 4.1 with the standard monotonic improvement bounds found in conservative methods like CPI (Kakade & Langford, 2002). In CPI, because the updated policy is a mixture, its expected advantage evaluates to a small fraction of the greedy advantage: $\mathbb{A}^{\pi_k}_{\pi_{k+1}} = \alpha \mathbb{E}_{s\sim d^{\pi_{k+1}}}[\max_a(q^{\pi_k}(s,a) - V^{\pi_k}(s))]$ (Agarwal et al., 2021), which restricts the update. We provide a more detailed comparison to the CPI result in Supplemental Materials B.

In contrast, Theorem 4.1 shows how SV-API can capture the full, unscaled expected advantage of its updated policy, $\mathbb{A}^{\pi_k}_{\pi_{k+1}}$. SV-API avoids scaling down the policy improvement because it does not explicitly set δ as a maximum target policy change. Rather, the algorithm waits for the behavioral policy to stabilize to ensure δ-NPA. If this occurs, δ is kept under control, but the estimated improvement $\mathbb{A}^{\pi_k}_{\pi_{k+1}}$ can be large because π_{k+1} is the accumulation of multiple training steps. In this way, a convergent sequence of behavioral policies avoids problematic distribution shift while permitting leaps in target policy performance. In fact, it is sufficient for the final behavioral policy, β_k, to have a bounded distance to β_{k+1}.

This is the motivation behind our proposed dynamic convergence criterion, which measures the value function change on the behavioral distribution. When the value network stops changing on a

Algorithm 3 Stable Value PPO

Require: Target $\pi_{\bar{\theta}}$, Value v_{ϕ_0}, Num. Env. Steps T,
Params: $\delta_v, K_{min}, K_{max}, \alpha_{ent}$
$\beta_{\theta_0} \leftarrow \pi_{\bar{\theta}}$, $k_\pi \leftarrow 0$, $n_{\text{stable}} \leftarrow 0$
for $k = 0, 1, 2, \ldots$ **do**
  $\mathcal{D}_k \leftarrow \{(s_t, a_t, s'_t, r_t)\}_{t=1:T}, a_t \sim \beta_{\theta_k}(s_t)$
  1. Evaluate Target Policy (`Eval`)
  $y_{1:T} \leftarrow \text{VTrace}(\mathcal{D}_k, \beta_{\theta_k}, \pi_{\bar{\theta}})$
  $\hat{A}_{1:T} \leftarrow \text{Retrace-GAE}(\mathcal{D}_k, \beta_{\theta_k}, \pi_{\bar{\theta}})$
  $\phi_{k+1} \leftarrow \text{Step}_\phi\left(\frac{1}{2}\sum_t (y_t - v_{\phi_k}(s_t))^2\right)$
  2. Update Behavioral Policy (Γ)
  $\theta_{k+1} \leftarrow \text{Step}_\theta(J^{clip}(\theta_k, \hat{A}_{1:T}))$
  3. Stability Criterion ($\mathcal{C}$)
  $\text{diff}_k \leftarrow \frac{1}{T}\sum_t |y_t - v_{\phi_k}(s_t)|$
  $\text{stable}, n_{\text{stable}} \leftarrow \texttt{Conv}(y_{1:T}, \text{diff}_k, n_{\text{stable}}, k_\pi)$
  if conv. **then**
    $k_\pi \leftarrow 0$
    $\bar{\theta} \leftarrow \theta_{k+1}$ ▷ Update target policy
  else
    k_π += 1 ▷ Hold target policy
  end if
end for

Algorithm 4 `Conv`

Require: $y_{1:T}$, diff_k, n_{stable}, k_π
In-Scope (Alg. 3): $\delta_v, K_{min}, K_{max}$
Check Stability:
$\overline{y}_T \leftarrow \frac{1}{T}\sum_{t=1}^{T} |y_t|$
$I_{\text{stable}} \leftarrow \mathbb{I}(\text{diff}_k / \overline{y}_T \leq \delta_v)$

Increment Number of Stable Iters:
$n_{\text{stable}} \leftarrow (n_{\text{stable}} + 1) \times I_{\text{stable}}$

Stability Gating / Early Stopping:
if $n_{\text{stable}} \geq K_{min}$ or $k_\pi \geq K_{max}$ **then**
  return True, 0
else
  return False, n_{stable}
end if

shifting distribution, we take this as a proxy for the underlying training process stabilizing – both the value estimate and resulting state distribution. If this is the case, then distribution shift is controlled by the stabilization of the data-collection process and value estimate, in addition to any conservative properties of Γ.

5 Empirical results

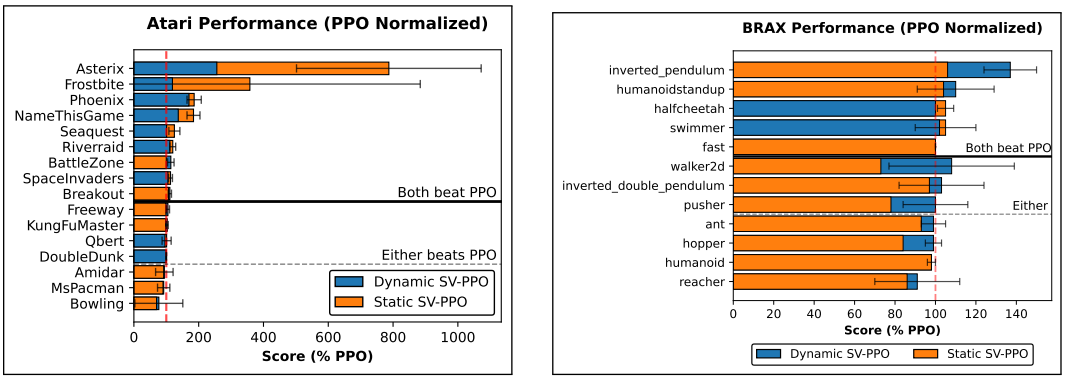
We implement Algorithm 2 by modifying the conservative deep RL method, PPO (Schulman et al., 2017). By preventing large changes in both the value function and (behavioral) policy, the clipped PPO update helps induce the stabilization dynamic described above. The resulting algorithm, SV-PPO, is shown in Algorithm 3. It maintains a frozen set of policy parameters, corresponding to the target policy. Since SV-PPO must estimate V^π and A^π from potentially off-policy data, standard importance sampling (IS) techniques with respect to the target policy are required. We use V-Trace (Espeholt et al., 2018) and ReTrace(λ) (Munos et al., 2016) for off-policy value and advantage estimation, respectively. Similar to TD(λ) Sutton (1988), V-Trace computes a return-based value target, but it uses clipped importance sampling ratios to reduce bias due to off-policy trajectories (full details are provided in Supplemental Materials P). The frozen policy action probabilities are used to compute importance sampling (IS) ratios.

To help ensure the distribution shift conditions of Theorem 4.1 hold, SV-PPO applies the clipping heuristic from PPO to subsequent *behavioral* policies within the standard PPO surrogate objective J^{clip}, which is provided in Supplemental Materials P. Standard PPO can be obtained from SV-PPO by ensuring $\mathcal{C}$ is always true, for example by setting $K_{max} = 1$ or $\delta_V = \infty$ and $K_{min} = 0$.

SV-PPO does not require any additional training or environment sampling beyond standard PPO. The primary additional computational cost is running inference with the frozen target policy, whose predictions are used in the IS ratio for VTrace and ReTrace(λ).

5.1 Atari

We run on sixteen Atari games: the Atari-10 Benchmark (Bellemare et al. (2013), Aitchison et al. (2022)), plus six well-known and interesting games: Breakout, Asterix, Freeway, Seaquest, Space Invaders, and Ms. Pacman. We run two variants of SV-PPO: One uses a dynamic convergence condition, and sets $\delta = 0.01$ and $K_{max} = 33$, the other uses a static convergence criterion of $K_{min} = K_{max} = 9$. Hyperparameters for $\mathcal{C}$ were obtained by tuning on MinAtar (Young & Tian, 2019), and besides these are due to PPO. We modify the hyperparameter for PPO compared to the original paper to help maintain performance on stochastic versions of the games (See Supplemental Materials 7 for full hyperparameters). Results are shown in Figure 4a.



(a) Atari benchmark suite. Five Seeds.

(b) BRAX continuous control suite. Twelve seeds.

Figure 4: **SV-PPO Performance Comparison.** Normalized scores relative to the PPO baseline (100%). Bars show the performance of both variants, with the colored tip indicating the winning variant and its margin.

Performance on each game is shown in Figure 4a. Dynamic SV-PPO saw better final performance on 11 out of the 16 games, and was statistically significantly better on six ($p = 0.1$, Welch's T-Test), and significantly worse on none. Static SV-PPO also never statistically significantly harmed performance, and improved performance on seven games.

Aggregate information about the size of the policy update is shown in Table 1. Both variants make fewer, larger updates to the target policy. The static interval variant made one ninth the number of target policy updates, each of which was 511% larger on average, as measured by the KL-divergence. Dynamic SV-PPO made anywhere from 3% to 95% as many updates, which were 96% larger on average. For per-game information, please reference Table 2 and 3 in the Supplemental Materials.

We also compare the difference in successive data collection policies *at the time of the target policy update* in the last column of Table 1. This indicates the degree of next policy alignment: The change in behavioral policy during the update step is reflected in the distribution shift terms in Theorem 4.1 and Equation 8. We find that SV-PPO, especially the dynamic variant, shrinks the distribution shift between policies *beyond the standard PPO clipping heuristic*. As there are no other differences between algorithms, this must be because SV-PPO's critic evaluates each policy for longer, helping stabilize the policy update.

We find that performance on most games benefited from slower target policy updates, which corresponds to more data and learning for each value function encountered in the API loop. Figures 5 and 6 in the Supplemental Materials show the convergence threshold for all games. The dynamic convergence criterion resulted in infrequent policy updates at the beginning (resembling policy iteration) and frequent refinements to the policy at the end (resembling value iteration). Based on this fact, along with its heuristic control of policy drift and value error terms that appear in our theory, we expected better performance from the dynamic variant. However, the static interval variant improved performance on most games where the stable value principle was helpful. This suggests

Table 1: Performance of Static and Dynamic SV-PPO across Brax and Atari environments (12 and 5 training seeds respectively). Scores are reported relative to PPO. The first column shows median performance, with the interquartile range in brackets. The next shows the geometric mean normalized performance and its multiplicative spread. The last two columns measure the mean change in target policy and behavioral policy during policy update steps, as measured by KL-divergence.

	Med. Score (% PPO)	**Mean**	**Beat PPO (%)**	**KL (π)**	**KL (β)**
Static (Brax)	97.4 [85.7, 104.0]	93.4 ($\times/\div$ 1.13)	42.0	14.89$\times$	1.40$\times$
Dynamic (Brax)	100.2 [98.5, 104.2]	103.2 ($\times/\div$ 1.11)	50.0	1.08$\times$	1.05$\times$
Static (Atari)	104.0 [98.8, 139.8]	133.3 ($\times/\div$ 1.84)	56.2	6.11$\times$	0.94$\times$
Dynamic (Atari)	105.0 [98.2, 115.2]	113.2 ($\times/\div$ 1.32)	68.8	1.96$\times$	0.70$\times$

that the convergence condition could potentially be improved. SV-PPO is able to shrink distribution shift, improve the value function estimate on the relevant distribution, and obtain a larger policy improvement.

5.2 Continuous Control

We test PPO and SV-PPO on standard continuous control benchmarks using the Brax physics simulator (Freeman et al., 2021). Final performance of Static and Dynamic SV-PPO relative to PPO is shown in Figure 4b. For PPO, the policy is a Gaussian, which we find leads to high variance importance sampling ratios. We hypothesize this offsets any benefits of regularizing value learning using the stable value principle. Indeed, it seems SV-PPO can degrade performance on environments where the PPO policy is highly deterministic, such as the `reacher` and `ant`. After initially experimenting with $K_{max} = 32$ as in Atari, we found better results with $K_{max} = 8$, which limits how off-policy the data can become. The static variant performs nearly as well as standard PPO with one eighth of the updates ($K_{max} = K_{min} = 8$). The dynamic thresholding variant set $\delta_v = 0.05$ and $K_{max} = 8$, and performed a median of 1% better across the suite. We speculate that applying Stable Value API to Soft Actor-Critic (SAC) Haarnoja et al. (2018) may help since SAC's entropy regularization may lower the variance of the IS ratios.

6 Conclusion

This work introduced ANPS, a new mechanism for policy improvement based on iteratively refining a behavioral policy until it is suitable to become the new target policy. We implement ANPS using SV-API, which waits until the change in the agent's value estimates falls below a threshold before updating the target policy. Our Theorem 4.1 demonstrates that if this heuristic stabilization corresponds to a bounded state-distribution shift between successive behavioral policies, monotonic policy improvement is guaranteed. Because SV-API enables significantly larger target policy updates while ensuring distribution shift does not degrade the expected return, it represents a fundamentally new approach to safe policy improvement.

Directions for future work include: (1) improving the empirical convergence criterion $\mathcal{C}$ to better approximate the unmeasurable theoretical quantities dictating alignment; (2) exploring alternative methods to execute ANPS, such as planning in imagination with a learned world model; (3) integrating principled exploration techniques directly into the behavioral policy update; and (4) investigating the practical impact of off-policy evaluation errors on SV-API and determining ways to reduce them.

Appendix

Proof of Theorem 3.3

Proof of Theorem 3.3. Start with the PDL, and introduce q^π through addition and subtraction:

$$\begin{aligned}(1-\gamma)(V^{\pi'}(s_0) - V^\pi(s_0)) &= \mathbb{E}_{s,a\sim d^{\pi'}}\ A^\pi(s,a) \quad \text{(PDL)} \\ &= \mathbb{E}_{s,a\sim d^{\pi'}}\ [q^\pi(s,a) - V^\pi(s) + Q^\pi(s,a) - q^\pi(s,a)] \\ &= \mathbb{A}^\pi_{\pi'} + \mathbb{E}_{s,a\sim d^{\pi'}}[Q^\pi(s,a) - q^\pi(s,a)]\end{aligned}$$

Subtract $\mathbb{A}^\pi_{\pi'}$ from both sides, and take the absolute value:

$$\begin{aligned}\left|(1-\gamma)(V^{\pi'}(s_0) - V^\pi(s_0)) - \mathbb{A}^\pi_{\pi'}\right| &= \left|\mathbb{E}_{s,a\sim d^{\pi'}}[Q^\pi(s,a) - q(s,a)]\right| \\ &\leq \mathbb{E}_{s,a\sim d^{\pi'}}|Q^\pi(s,a) - q(s,a)| \\ &= \varepsilon(d^{\pi'}, q^\pi)\end{aligned}$$

Dividing by $(1-\gamma) > 0$ and noting $|x| \leq a \iff -a \leq x \leq a$ yields the theorem statement.

$$\begin{aligned}&-\varepsilon(d^{\pi'}, q^\pi) \leq (1-\gamma)(V^{\pi'}(s_0) - V^\pi(s_0)) - \mathbb{A}^\pi_{\pi'} \leq \varepsilon(d^{\pi'}, q^\pi) \\ &\frac{1}{1-\gamma}\left(\mathbb{A}^\pi_{\pi'} - \varepsilon(d^{\pi'}, q^\pi)\right) \leq (V^{\pi'}(s_0) - V^\pi(s_0)) \leq \frac{1}{1-\gamma}\left(\mathbb{A}^\pi_{\pi'} + \varepsilon(d^{\pi'}, q^\pi)\right)\end{aligned}$$

□

Acknowledgments

This material is based upon work supported by ARO grant #W911NF2210251. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ARO.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation, 2018. URL https://arxiv.org/abs/1806.06920.

Alekh Agarwal, Nan Jiang, Sham M. Kakade, and Wen Sun. *Reinforcement Learning: Theory and Algorithms*. 2021. URL https://rltheorybook.github.io/.

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games, 2022. URL https://arxiv.org/abs/2210.02019.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996. ISBN 978-1886529106.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1etN1rtPB.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL https://arxiv.org/abs/1802.01561.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7IzeL0kflu.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://dblp.uni-trier.de/db/conf/icml/icml2018.html#HaarnojaZAL18.

Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. Muesli: Combining improvements in policy optimization, 2022. URL https://arxiv.org/abs/2104.06159.

R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. *CoRR*, abs/1606.02647, 2016. URL http://arxiv.org/abs/1606.02647.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL https://arxiv.org/abs/2303.08774.

Theodore Perkins and Doina Precup. A convergent form of approximate policy iteration. In S. Becker, S. Thrun, and K. Obermayer (eds.), *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/9f44e956e3a2b7b5598c625fcc802c36-Paper.pdf.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2020. URL http://arxiv.org/abs/2009.01325. cite arxiv:2009.01325Comment: NeurIPS 2020.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3 (1):9–44, August 1988. ISSN 0885-6125. DOI: 10.1023/A:1022633531479. URL https://doi.org/10.1023/A:1022633531479.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999. URL http://dblp.uni-trier.de/db/journals/ai/ai112.html#SuttonPS99.

Hongyao Tang and Glen Berseth. Improving deep reinforcement learning by reducing the chain effect of value and policy churn, 2024. URL https://arxiv.org/abs/2409.04792.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Proof of Lemma 3.4

Given two distributions, μ, and arbitary action-value estimate q^π, we'd like to $\varepsilon(\mu' q^\pi)$. Dropping dependence on q^π, and writing the difference:

$$\varepsilon(\mu') - \varepsilon(\mu)$$

Since $\varepsilon(\mu)$ is positive for any distribution, we have

$$\begin{aligned}
\varepsilon(\mu') - \varepsilon(\mu) &\leq |\varepsilon(\mu') - \varepsilon(\mu)| && (11)\\
&= |\mathbb{E}_{s,a\sim\mu'}|Q^\pi(s,a) - q^\pi(s,a)| - \mathbb{E}_{s,a\sim\mu}|Q^\pi(s,a) - q^\pi(s,a)|| && (12)\\
&= \left|\sum_{s,a}(\mu' - \mu)|Q^\pi(s,a) - q^\pi(s,a)|\right| && (13)\\
&\leq \left|\sum_{s,a}|\mu' - \mu||Q^\pi(s,a) - q^\pi(s,a)|\right| && (14)\\
&= \sum_{s,a}|\mu' - \mu||Q^\pi(s,a) - q^\pi(s,a)| && (15)\\
&\leq \max_{s,a,\pi}(|Q^\pi(s,a) - q^\pi(s,a)|)\sum_{s,a}|\mu' - \mu| && (16)\\
&\leq \frac{1}{1-\gamma}\sum_{s,a}|\mu' - \mu| && (17)\\
&= \frac{2}{1-\gamma}d_{TV}(\mu', \mu) && (18)
\end{aligned}$$

Rearranging gives

$$\varepsilon(\mu') \leq \varepsilon(\mu) + \frac{2}{1-\gamma}d_{TV}(\mu', \mu) \quad (19)$$

B Comparison of Theorem 4.1 to CPI Monotonic Policy Improvement Bound

We compare our Theorem 3.3 to the related monotonic policy improvement bound from CPI, which we state below. Note that the following bound uses ϵ that is distinct from $\varepsilon(\mu, q^\pi)$. CPI uses ϵ to denote error in the greedy policy selection subroutine. Recall that in our notation, $\varepsilon(q^{\pi_k}, \mu)$ measures the explicit mean value estimation error of the critic on its distribution μ. These terms are related, but are not interchangeable.

Theorem B.1 (Agarwal et al. (2021) Theorem 12.2)**.** *Assume access to a method which outputs a policy π' such that $\mathbb{A}_{\pi'}^{\pi_k} \geq \max_{\tilde{\pi}} \mathbb{A}_{\tilde{\pi}}^{\pi_k} - \epsilon$. Then, the CPI update $\pi_{k+1} = (1-\delta(1-\gamma))\pi_k + \delta(1-\gamma)\pi'$ yields the following performance bound:*

$$V^{\pi_{k+1}}(s_0) - V^{\pi_k}(s_0) \geq \frac{1}{1-\gamma}\left(\delta(1-\gamma)[\max_{\tilde{\pi}} \mathbb{A}_{\tilde{\pi}}^{\pi_k} - \epsilon] - 2\delta^2\gamma\right) \tag{20}$$

Note that the distribution shift penalty in the last term is of $O(\delta^2/(1-\gamma))$, while for SV-API it is of $O(\delta/(1-\gamma)^2)$. This is because, for CPI, the coefficient $\delta(1-\gamma)$ scales down both the expected advantage and distribution shift. This allows CPI to guarantee monotonic policy improvement for small enough δ. In contrast, the bound in Theorem 4.1 is due to an unconstrained policy update, and incurs no multiplier of δ on the improvement $\mathbb{A}_{\pi_{k+1}}^{\pi_k}$.

C Detailed Atari Convergence Threshold

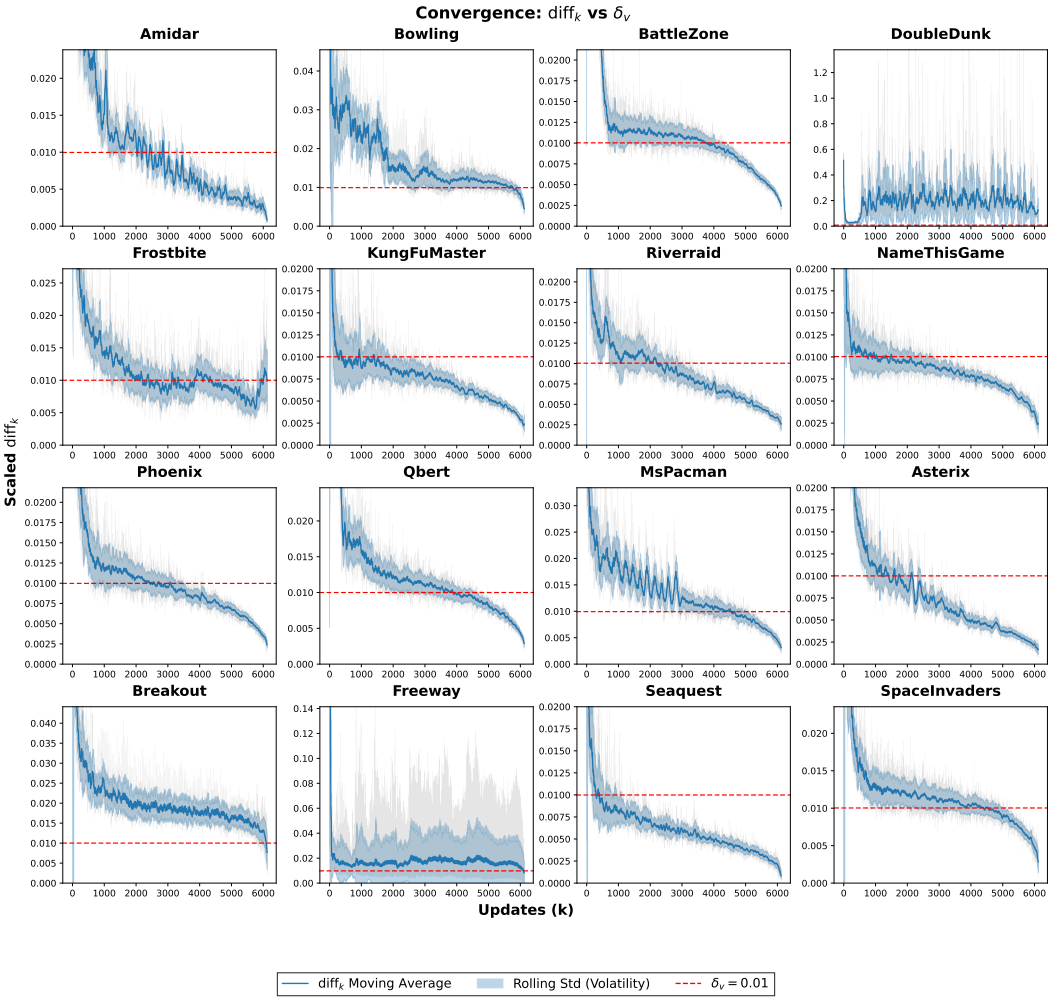


Figure 5: Scaled value diff and convergence threshold δ_v for SV-PPO on all 16 Atari games.

D Detailed Atari Learning Curves

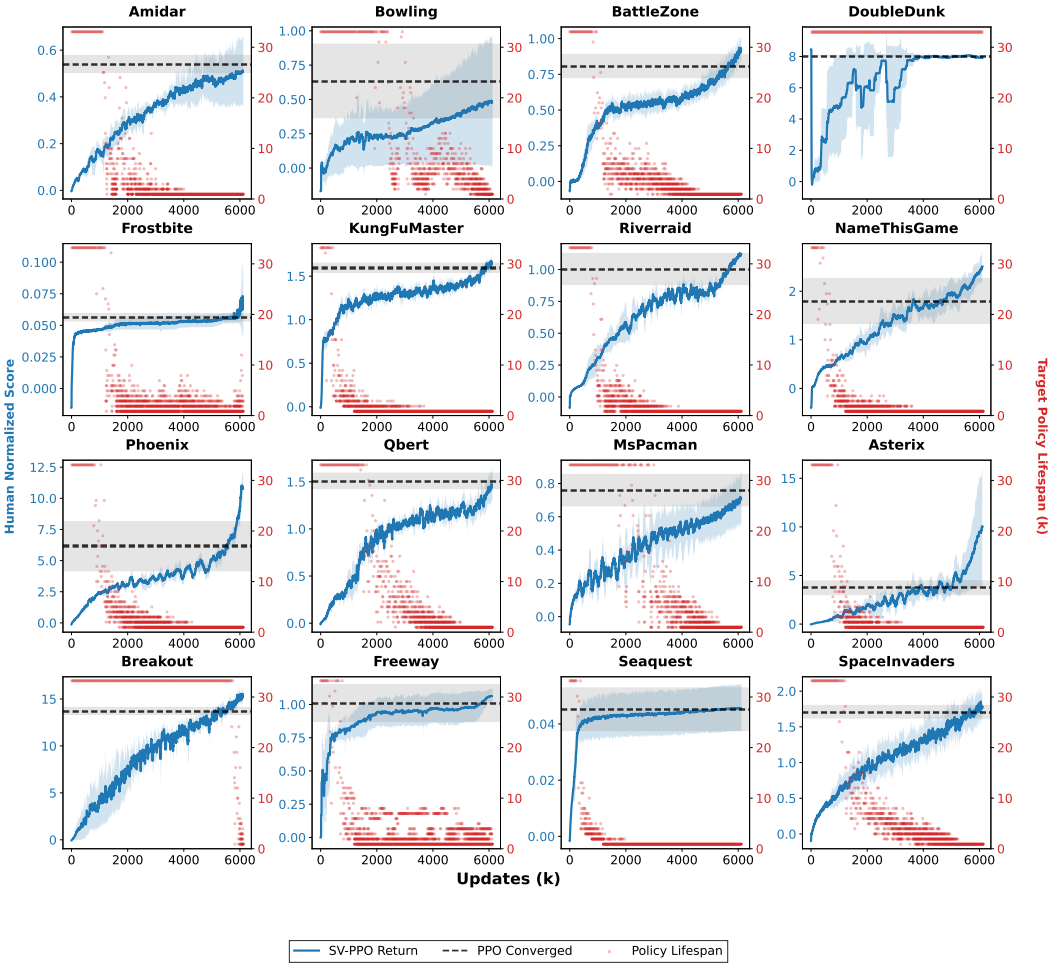


Figure 6: Average score and target policy lifespan for SV-PPO on all 16 Atari games.

E Learning Curves: Atari

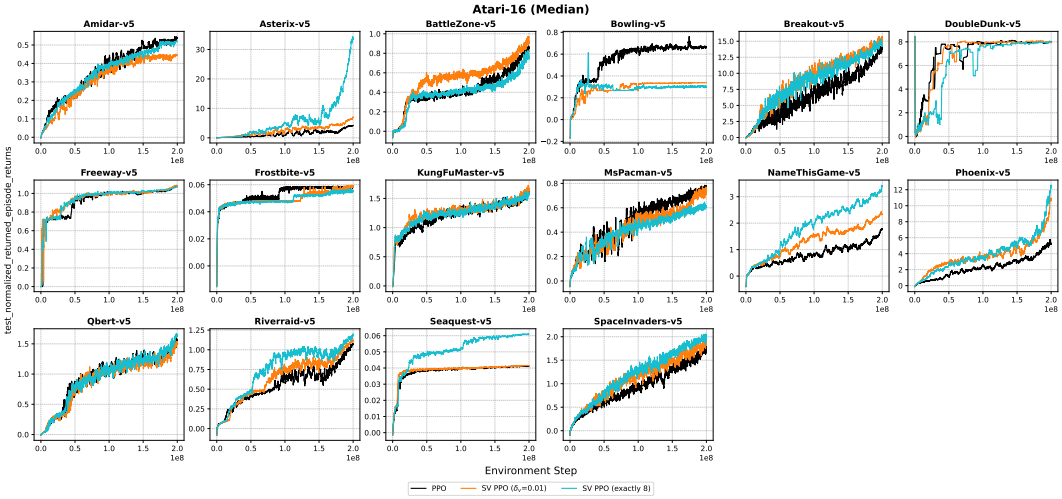
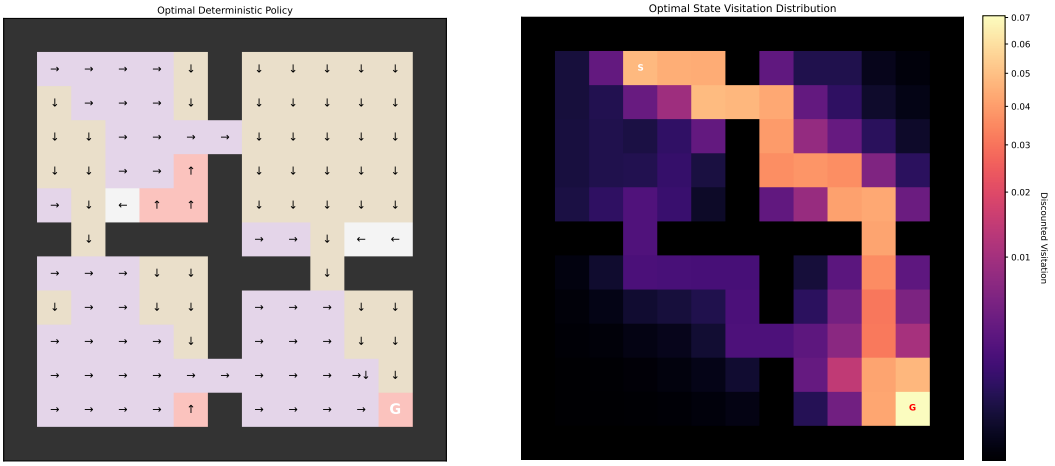


Figure 7: Learning Curves for PPO, Dynamic SV-PPO, and Static SV-PPO for Atari (5 seeds, median shown 0.

F Four Rooms Experiment

Four Rooms (Sutton et al., 1999) is a grid world with fixed starting state (top left room) and goal state (bottom right room) The action is the intended direction of movement, and is executed noisily, with a 20% chance of success. Hugging the walls mitigates the negative effect of an erroneous move. The optimal policy, shown in Figure (8a), visits all four rooms: generally going through the top room, but occasionally going through the bottom room if brought there via noise.

We use 10,000 rounds of value iteration to solve for the optimal policy on Four Rooms. We then compute the optimal actions, placing equal weight on ties, and compute the associated discounted stationary distribution.



(a) Optimal Policy

(b) Associated State Visitation Distribution

Figure 8: Optimal Policies and Associated State Visitation Distribution

F.1 Experiment Details

We run an implementation of PPO due to Lu et al. (2022), using a convolutional actor critic network, due to Gallici et al. (2025)). The observation space is visual (the entire grid, with the agent's position and wall marked). SV-PPO uses dynamic thresholding, and is discussed in detail in the next section. We run for 195 total rounds, each corresponding to an update to the behavioral policy. Based on the strictness of the convergence criterion, SV-PPO updates the target policy 40 times. Figure 3 visualizes the convergence logic (corresponding to $\mathcal{C}$ in Algorithm 2): If the quantity shown in blue on the rightmost chart falls below the dotted line for four consecutive iterations, an update occurs. We use $\lambda = 0.8$ for the GAE(λ) and TD(λ) targets.

F.2 Results

In terms of performance, SV-PPO converges to the optimal policy. PPO has unstable convergence to a local optimum that always attempts to go through the top rooms even if noise has pushed the agent closer to the bottom-left room, while the optimal policy is to continue along the bottom route. Of note is the sudden collapse in the performance of PPO in iteration $k = 80$. This is due to mis-generalization of the value network, sometimes called value churn (Tang & Berseth, 2024). Around iteration $k = 50$, PPO's CNN critic over-estimates the value along the left wall. By $k = 60$, the PPO policy network learns to avoid the bottom room. Insufficient sampling and poor generalization leads to catastrophic forgetting occurring at iteration $k = 90$. This is indicated by a massive degradation in the true value function at these states. The "recovery" in value that occurs for PPO is actually to

avoid these states, which is a sub-optimal policy. Detailed visualizations of the learning dynamics are in Figures 9 and 10.

By evaluating a fixed policy for longer, SV-PPO mitigated value overestimation. For example, it experienced less-overestimation along the left wall, since it was fitting a lower value policy in the top right room. Lower value estimates led to more consistent exploration of the problematic room – instead of just seeking the left wall, as PPO did. The end result is that SV-PPO learns a strong policy in the bottom left room, while PPO's remains weak, even at the end of learning.

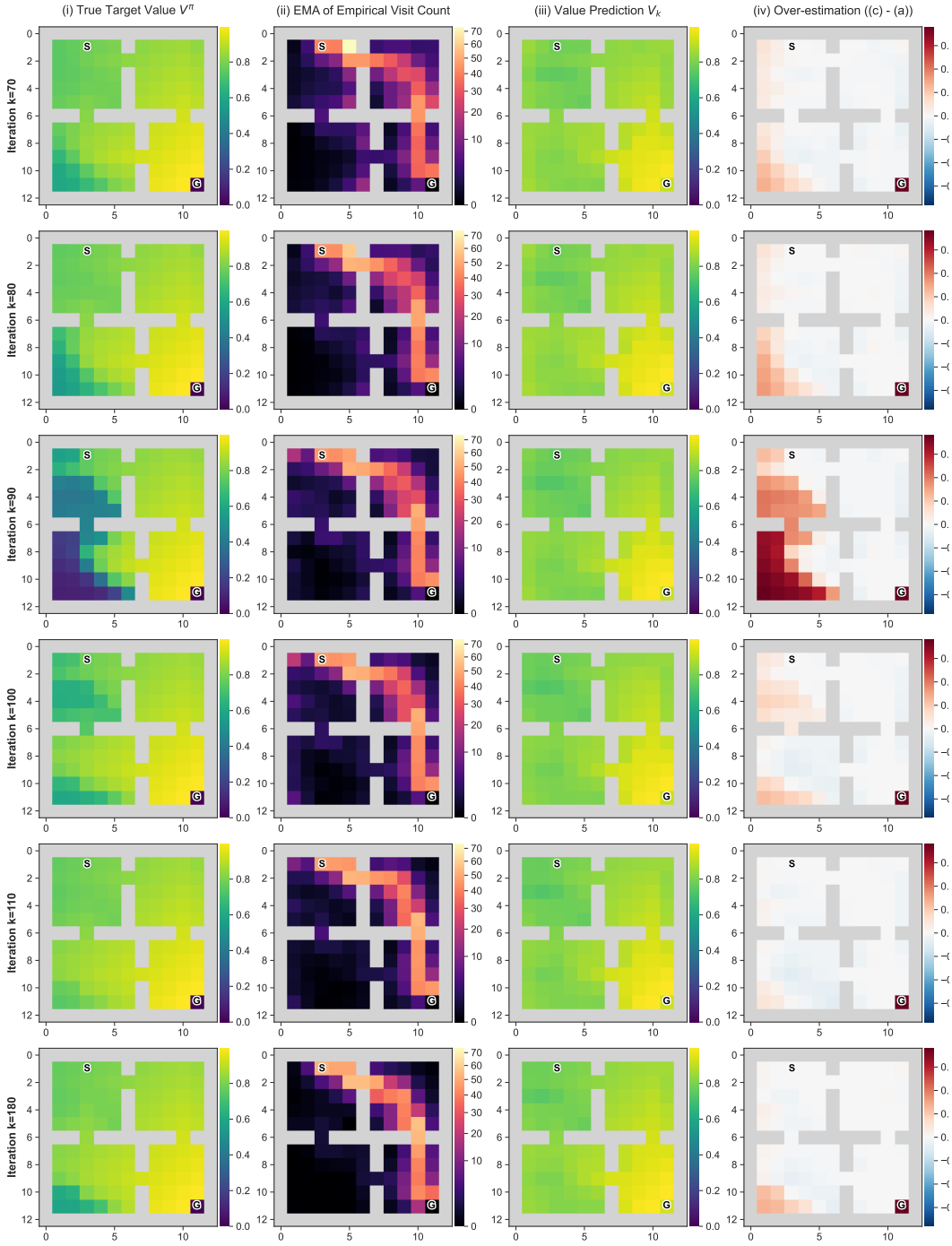


Figure 9: Detailed Examination of PPO's Learning on Four Rooms, highlighting the catastrophic forgetting at iteration 90, and over exploration of the bottom left corner due to value over -estimation at Iteration 100. This leads the agent to avoid the room altogether.

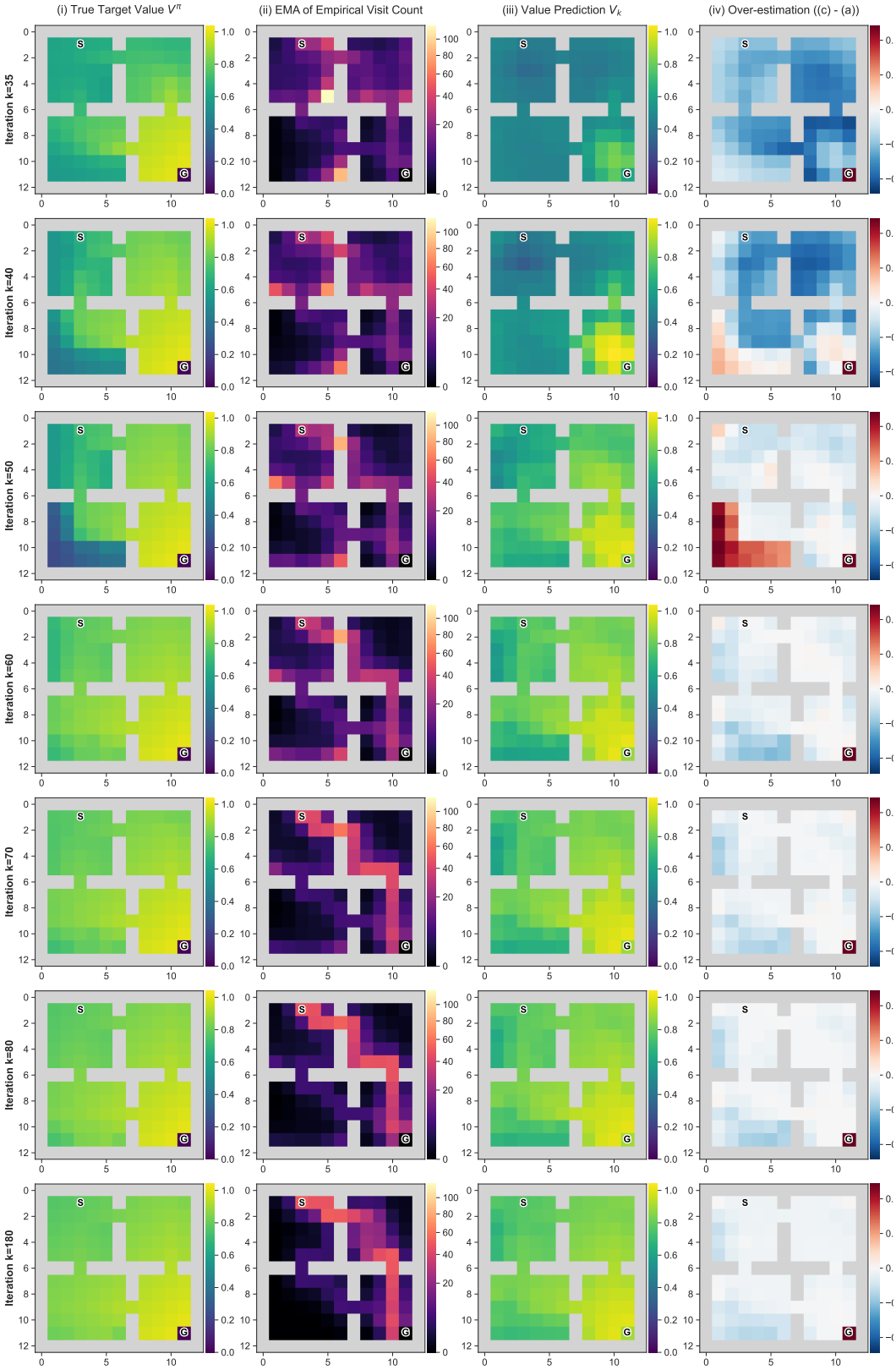


Figure 10: Detailed Examination of SV-PPO's Learning on Four Rooms, highlighting iterations which saw value over-estimation the bottom left corner, but less widespread than PPO, and more quickly resolved by exploring the room. The final policy goes through the bottom room just like the optimal policy in Figure (8)

G Static SV-PPO Atari

Table 2: **Static SV-PPO on Atari.** By strictly gathering data for 9 rounds before every target policy update, SV-PPO operates on 11% of the target updates of standard PPO as shown in **(b) Updates**. **Column (a)** indicates the final converged return relative to PPO. **Column (b)** indicates the fraction of rounds k where SV-PPO updates the target policy. **Column (c)** shows the size of the target policy jump as a multiplier of PPO's, demonstrating that SV-PPO executes substantially larger target updates. **Column (d)** shows the same quantity for the behavioral policy remains constrained.

Environment	(a) Score (% of PPO)	(b) Updates (% of PPO)	(c) $\text{KL}(\pi_k \| \pi_{k+1})$ ($\times$ PPO)	(d) $\text{KL}(\beta_k \| \beta_{k+1})$ ($\times$ PPO)
Asterix	**787 ± 285**	11	4.5x	0.7x
Frostbite	358 ± 526	11	7.6x	1.1x
Phoenix	**186 ± 22**	11	4.5x	0.8x
NameThisGame	**184 ± 20**	11	5.3x	1.0x
Seaquest	**125 ± 17**	11	6.9x	1.1x
Riverraid	**120 ± 9**	11	7.3x	1.0x
SpaceInvaders	**113 ± 6**	11	4.7x	0.9x
Breakout	**107 ± 3**	11	6.2x	0.9x
Qbert	101 ± 14	11	7.5x	0.9x
DoubleDunk	100 ± 1	11	11.3x	2.0x
BattleZone	100 ± 11	11	6.6x	0.9x
Freeway	99 ± 14	11	4.9x	1.0x
KungFuMaster	98 ± 8	11	6.1x	0.9x
Amidar	92 ± 10	11	5.7x	0.9x
MsPacman	90 ± 14	11	6.7x	0.9x
Bowling	70 ± 32	11	4.9x	0.6x

H Dynamic SV-PPO Atari

Table 3: **Dynamic SV-PPO on Atari 2600 environments. Column (a)** indicates the final converged return relative to PPO, with bold indicating significantly different performance. **Column (b)** indicates the fraction of rounds k where SV-PPO updates the target policy. **Columns (c) and (d)** show statistics about the convergence value ($\text{diff}_k/\overline{y}_T$ in Algorithm 4), normlaized by the threshold δ_v, such that a value ≤ 1 implies stability.**Column (e)** shows the size of the target policy jump as a multiplier of PPO's, demonstrating that SV-PPO executes substantially larger target updates. **Column (f)** shows the same quantity for the behavioral policy.

Environment	(a) **Score** (% of PPO)	(b) **Updates** (% of PPO)	(c) $\textbf{diff}_k/\delta_v$ (Median)	(d) **Std. diff**$_k$ (% of δ_v)	(e) $\textbf{KL}(\pi_k \| \pi_{k+1})$ (x PPO)	(f) $\textbf{KL}(\beta_k \| \beta_{k+1})$ (x PPO)
Asterix	**256 ± 151**	74	56	27	0.7x	0.5x
Phoenix	**170 ± 16**	61	85	25	0.9x	0.5x
NameThisGame	**137 ± 16**	75	75	21	1.3x	1.1x
Frostbite	119 ± 40	55	83	37	1.3x	0.6x
BattleZone	**114 ± 10**	47	96	27	1.8x	0.8x
Riverraid	**111 ± 1**	67	73	27	1.2x	0.7x
Breakout	**111 ± 5**	4	187	90	9.6x	0.7x
Freeway	105 ± 5	38	106	189	2.3x	0.9x
SpaceInvaders	105 ± 12	41	100	24	1.4x	0.6x
KungFuMaster	103 ± 3	79	67	21	1.2x	1.0x
Seaquest	101 ± 18	85	47	18	1.2x	1.0x
DoubleDunk	99 ± 1	3	1645	3299	27.0x	0.9x
Qbert	96 ± 11	40	100	28	1.9x	0.6x
Amidar	94 ± 27	54	83	36	1.3x	0.6x
MsPacman	92 ± 19	27	115	39	2.8x	0.4x
Bowling	77 ± 74	40	105	2865	2.4x	0.7x

I Dynamic SV-PPO Brax

Table 4: **Dynamic SV-PPO on Continuous Control Column (a)** indicates the final converged return relative to PPO. **Column(b)** indicates the fraction of rounds k where SV-PPO updates the target policy. **Columns (c) and (d)** show statistics about the convergence value ($\text{diff}_k/\overline{y}_T$ in Algorithm 4), normalized by the threshold δ_v, such that a value ≤ 1 implies stability. **Column(e)** indicates value loss (MSBE) at convergence as a multiplier of PPO's. **Column (f)** shows the size of the target policy jump as a multiplier of PPO's, demonstrating that SV-PPO executes substantially larger target updates. **Column (g)** shows the same quantity for the behavioral policy. **Column (h)** shows mean policy entropy of PPO at convergence.

Environment	(a) **Score** (% of PPO)	(b) **Updates** (% of PPO)	(c) $\textbf{diff}_k/\delta_v$ (Median)	(d) **Std. diff**$_k$ (% of δ_v)	(e) **Value Loss** (x PPO)	(f) $\textbf{KL}(\pi_k\|\|\pi_{k+1})$ (x PPO)	(g) $\textbf{KL}(\beta_k\|\|\pi_{k+1})$ (x PPO)	(h) $\boldsymbol{H}(\pi_k)$ (PPO)
inverted_pendulum	137 ± 13	97.11	45	72	2.12x	1.0x	1.0x	-2.39
humanoidstandup	110 ± 19	97.12	46	20	1.03x	1.0x	1.0x	36.89
walker2d	108 ± 31	97.23	108	37	0.93x	1.0x	1.0x	1.82
inverted_double_pendulum	103 ± 21	97.24	35	28	0.97x	1.2x	1.2x	-0.43
swimmer	102 ± 3	96.91	46	24	1.08x	1.2x	1.1x	2.53
halfcheetah	100 ± 3	96.92	36	14	0.99x	1.1x	1.0x	5.24
fast	100 ± 0	97.22	9	6	0.99x	1.1x	1.0x	-0.23
pusher	100 ± 16	97.03	63	46	0.83x	1.3x	1.2x	-14.62
hopper	99 ± 4	97.20	51	24	0.80x	1.1x	1.1x	3.95
ant	99 ± 6	93.82	67	38	1.49x	1.0x	1.0x	-0.07
humanoid	98 ± 2	97.18	48	21	1.08x	1.0x	1.0x	28.95
reacher	91 ± 21	97.20	83	37	0.89x	1.0x	1.0x	-7.16

J Static SV-PPO Brax

Table 5: **Static SV-PPO on Continuous Control. Column (a)** indicates the final converged return relative to PPO. **Column(b)** indicates the fraction of rounds k where SV-PPO updates the target policy. **Column(c)** indicates value loss (MSBE) at convergence as a multiplier of PPO's. **Column (d)** shows the size of the target policy jump as a multiplier of PPO's, demonstrating that SV-PPO executes substantially larger target updates. **Column (e)** shows the same quantity for the behavioral policy. **Column (f)** shows mean policy entropy of PPO at convergence.

Environment	(a) Score (% of PPO)	(b) Updates (% of PPO)	(c) **Value Loss** (x PPO)	(d) $\mathrm{KL}(\pi_k \| \pi_{k+1})$ (x PPO)	(e) $\mathrm{KL}(\beta_k \| \pi_{k+1})$ (x PPO)	(f) $\boldsymbol{H(\pi_k)}$ (PPO)
inverted_pendulum	106 ± 49	12.49	2.17x	5.0x	0.5x	-2.39
halfcheetah	105 ± 4	12.49	0.84x	7.9x	1.3x	5.24
swimmer	105 ± 15	12.49	0.90x	12.7x	1.1x	2.53
humanoidstandup	104 ± 11	12.49	0.86x	15.5x	1.2x	36.89
fast	100 ± 0	12.49	1.06x	40.1x	1.8x	-0.23
humanoid	98 ± 5	12.49	0.98x	9.7x	1.2x	28.95
inverted_double_pendulum	97 ± 20	12.49	0.59x	16.1x	1.7x	-0.43
ant	93 ± 8	12.49	1.23x	7.9x	1.1x	-0.07
reacher	86 ± 11	12.49	0.84x	11.7x	1.0x	-7.16
hopper	84 ± 13	12.49	0.65x	20.6x	1.9x	3.95
pusher	78 ± 16	12.49	0.91x	72.0x	7.0x	-14.62
walker2d	73 ± 19	12.49	1.01x	17.8x	1.2x	1.82

K Game-By-Game Comparison: Atari

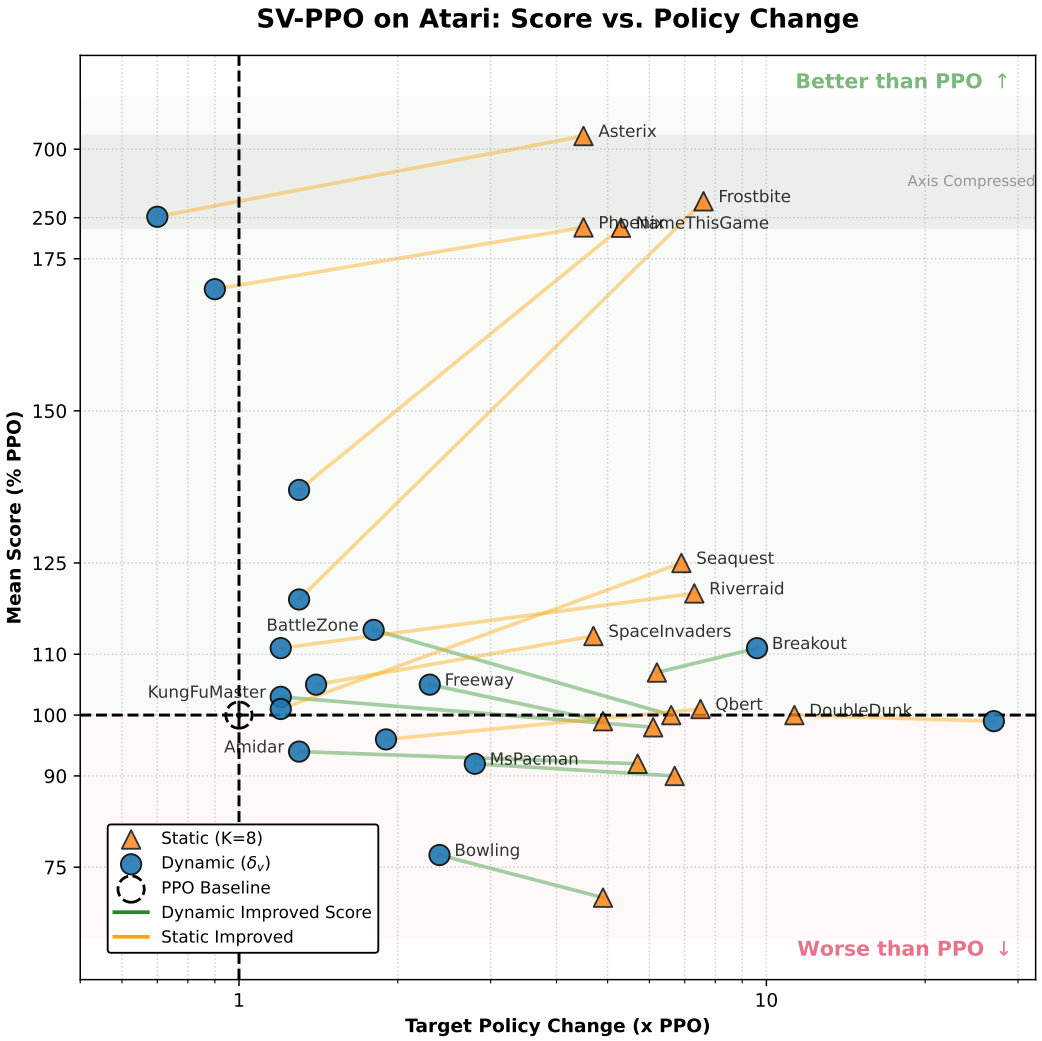


Figure 11: Game by Game comparison of Static and Dynamic SV-PPO on Atari. Each game is placed on the plot twice: a blue circle denotes the dynamic variant, and a triangle denotes the static variant. A colored line connects the same game for the two variants. The vertical axis is converged performance (normalized by PPO's) and the horizontal axis is the average change to the target policy, as measured by KL-divergence. Thus, if the line slopes "upward" for a certain game, this indicates the static variant increased performance. Moreover, the origin indicates PPO by construction.

L Game-By-Game Comparison: Brax

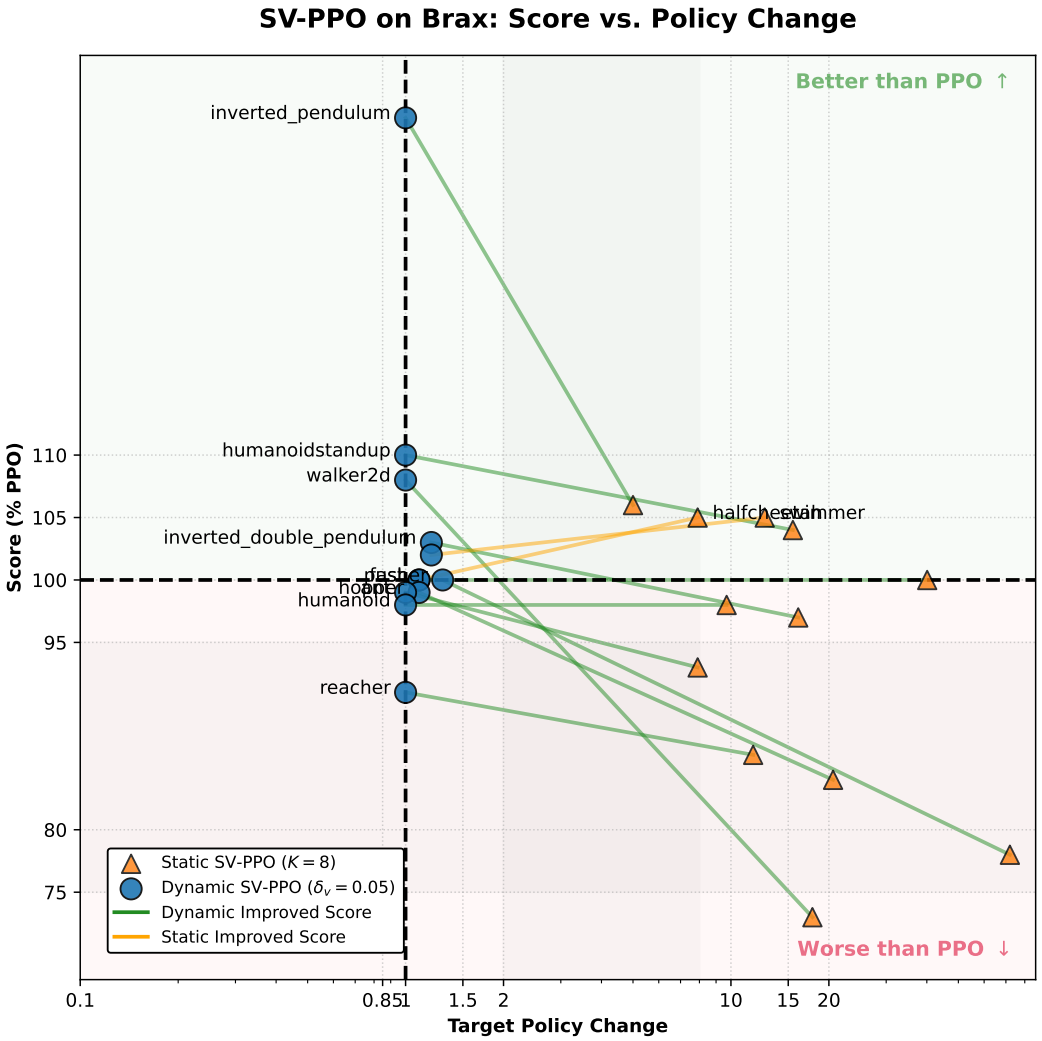


Figure 12: Game by Game comparison of Static and Dynamic SV-PPO on Brax. Each game is placed on the plot twice: a blue circle denotes the dynamic variant, and a triangle denotes the static variant. A colored line connects the same game for the two variants. The vertical axis is converged performance (normalized by PPO's) and the horizontal axis is the average change to the target policy, as measured by KL-divergence. Thus, if the line slopes "upward" for a certain game, this indicates the static variant increased performance. Moreover, the origin indicates PPO by construction.

M Learning Curves: Brax

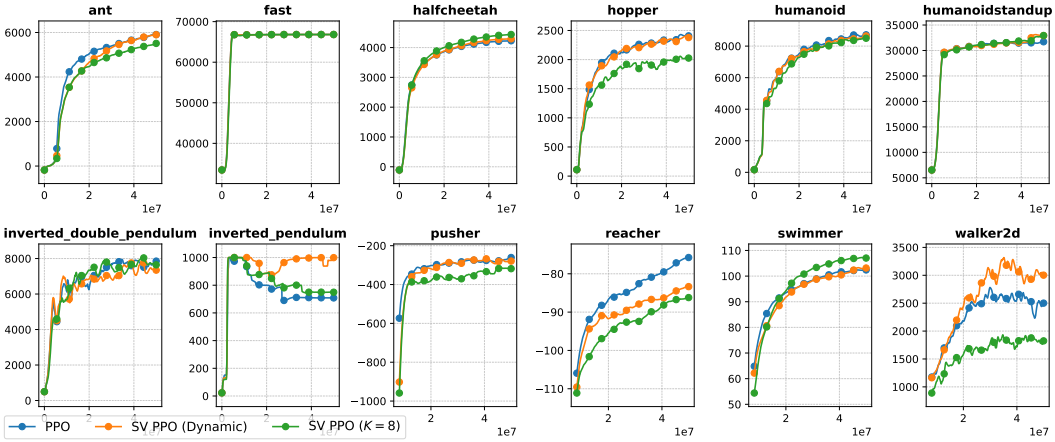


Figure 13: Learning Curves for PPO, Dynamic SV-PPO, and Static SV-PPO for Brax.

N SV-PPO Brax Hyperparameters

Table 6: **Hyperparameters for Brax Continuous Control.** Shared parameters (due to the Pure-JAXRL repository Lu et al. (2022)) are applied across all baselines and variants. SV-PPO specific parameters apply only to the dynamic and static gated algorithms.

Category	Hyperparameter	Value
Shared PPO Hyperparameters		
Environment & Sampling	Total Timesteps	5×10^7
	Number of Environments	2048
	Steps per Environment (T)	10
	Discount Factor (γ)	0.99
	GAE Parameter (λ)	0.95
Optimization (Network)	Learning Rate (Initial)	3×10^{-4}
	Learning Rate (Final)	1×10^{-5}
	LR Schedule	Linear
	Optimization Epochs	4
	Minibatch Size	1024
	Activation Function	Tanh
	Max Gradient Norm	1.0
	Initial Log Std	0.5
	Normalize Observations	True
	Normalize Rewards	True
	Warmup Steps	1000
Loss Coefficients	PPO Clip Range (ϵ)	0.2
	Value Clip Range	0.2
	Value Coefficient (c_2)	0.5
	Entropy Coefficient (α_{ent})	0.001
Dynamic SV-PPO Specific Hyperparameters		
Stability Gating	Value Diff Threshold (δ_v)	0.05
	Min Stable Iters (K_{min})	2
	Max Policy Delay (K_{max})	8
	IS Ratio Bounds ($\bar{c}, \bar{\rho}$)	$[0.0, 100.0]$
	K Decay Fraction	0.05
Static SV-PPO Specific Hyperparameters		
	Min Stable Iters (K_{min})	8
	Max Policy Delay (K_{max})	8
	IS Ratio Bounds ($\bar{c}, \bar{\rho}$)	$[0.0, 100.0]$

O SV-PPO Atari Implementation and Hyperparameters

Table 7: **Hyperparameters for Atari-16.** The PPO implementation is due to PureJAXRL, with most hyperparameters due to the original paper Schulman et al. (2017). However, following Museli's Hessel et al. (2022) strong implementation of PPO *with sticky actions*, we made several changes. First, separate IMPALA networks were used for actor and critic. The discount factor was raised to $\gamma = 0.995$ and the maximum learning rate to 3×10^{-4}. The minibatch size was also increased to 1,024. These changes improved performance on all tested games, and were shared by both runs. The SV-PPO stability gating hyperparameters were tuned on the MinAtar environments Young & Tian (2019), which can be run much more quickly than experiments on Atari. SV-PPO specific parameters apply only to the dynamic and static gated algorithms.

Category	Hyperparameter	Value
Shared PPO Hyperparameters		
Environment & Sampling	Total Timesteps	$200M$
	Number of Environments	64
	Steps per Environment (T)	128
	Discount Factor (γ)	0.995
	GAE Parameter (λ)	0.95
	Frame Skip	4
	No-op Max	30
	Repeat Action Probability	0.25
	Episodic Life	True
	Reward Clipping	False
Optimization (Network)	Learning Rate (Initial)	3×10^{-4}
	Learning Rate (Final)	1×10^{-6}
	LR Schedule	Linear
	Optimization Epochs	4
	Minibatch Size	1024
	Max Gradient Norm	1.0
	Layer Normalization	True
Loss Coefficients	PPO Clip Range (ϵ)	0.1
	Value Clip Range	0.2
	Value Coefficient (c_2)	0.25
	Entropy Coefficient (α_{ent})	0.01
Dynamic SV-PPO Specific Hyperparameters		
Stability Gating	Value Diff Threshold (δ_v)	0.01
	K_{min} Decay Fraction	0.2
	Min Stable Iters (K_{min})	9 decays to 1
	Max Policy Delay (K_{max})	33
	IS Ratio Bounds ($\bar{c}, \bar{\rho}$)	$[0.0, 5.0]$
Static SV-PPO Specific Hyperparameters		
Stability Gating	Min Stable Iters (K_{min})	9
	Max Policy Delay (K_{max})	9
	IS Ratio Bounds ($\bar{c}, \bar{\rho}$)	$[0.0, 5.0]$

P SV-PPO

Let π_θ denote the policy actor network, with θ_k parameterizing the current behavioral policy at round k, and $\overline{\theta}_k$ denoting the target policy parameters. The PPO policy optimization objective stays the same, except that the ratio $r_k(\theta)$ is the ratio of behavioral policies.

$$J^{clip}(\theta, \hat{A}_{1:T}) = \hat{\mathbb{E}}_t \left[\min \left(\frac{\beta_\theta(a_t|s_t)}{\beta_{\theta_k}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\beta_\theta(a_t|s_t)}{\beta_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \tag{21}$$

Optionally, entropy regularization can be added. In this way, the behavioral policy is allowed to drift further and further away from the target policy in each round. Once the convergence criterion is met, the update is just $\overline{\theta} \leftarrow \theta_{k+1}$.

The critic training process must ensure that the value function's fixed point corresponds to $\pi_{\overline{\theta}_k}$. Let the value network be v_ϕ. We use the clipped importance sampling method V-Trace Espeholt et al. (2018), which can be written in recursive terms:

$$\rho_t = \min \left(\bar{\rho}, \frac{\pi_{\overline{\theta}_k}(a_t|x_t)}{\pi_{\theta_k}(a_t|x_t)} \right), \quad c_i = \min \left(\lambda, \frac{\pi_{\overline{\theta}_k}(a_i|x_i)}{\pi_{\theta_k}(a_i|x_i)} \right) \tag{22}$$

$$\begin{aligned} \delta_t &= r_t + \gamma v_\phi(s_{t+1}) - v_\phi(s_t) \\ y_t &= v_\phi(s_t) + \rho_t \delta_t + \gamma c_t (y_{t+1} - v_\phi(s_{t+1})) \\ &\text{with} \quad y_{T+1} = v_\phi(s_{T+1}) \end{aligned} \tag{23}$$

Since c_t is at most λ, this results in TD(λ) when on-policy ($\theta_k = \overline{\theta}_k$). To eliminate bias due to off policy sampling, $\overline{\rho}$ must be set to infinity. Clipping with $\overline{\rho} < \infty$ stabilizes learning at the cost of adding some bias. We set $\overline{\rho}$ to 5. Since the advantage estimates are also off-policy, we modify the generalized advantage estimation (Schulman et al., 2018) used by PPO by incorporating the importance sampling weights from Equation 22, which are originally due to Retrace(λ) (Munos et al., 2016).

$$\hat{A}_t = \delta_t + \gamma c_t \hat{A}_{t+1}, \quad \hat{A}_{T+1} = 0 \tag{24}$$

Finally, we normalize the estimates $\hat{A}_t$ by dividing by the batch standard deviation, but we do not center them, to avoid flipping the sign.