# Reliable Policy Iteration: Consistent Performance across Function Approximators

S.R. Eshwar[1], Kintan Saha[2], Aniruddha Mukherjee[3], Krishna Agarwal[2],
Gugan Thoppe[1], Aditya Gopalan[4], Gal Dalal[5]

***Abstract*— Policy Iteration (PI) is a cornerstone of reinforcement learning, yet its fundamental guarantee of monotonic improvement collapses when combined with function approximation, leading to instability in modern algorithms. Reliable Policy Iteration (RPI) was recently proposed to address this by reformulating policy evaluation as a constrained optimization problem, provably restoring monotonic improvement guarantees. In this work, we move beyond theory and conduct a rigorous empirical evaluation of RPI's resilience to the choice of function approximator. We systematically vary the capacity of both linear and neural network-based approximators across a suite of control tasks, from model-based inventory control to model-free deep reinforcement learning in CartPole and Inverted Pendulum. Our experiments consistently demonstrate that RPI and its model-free variants, $\text{RPI}_{\text{DQN}}$ and $\text{RPI}_{\text{DDPG}}$, significantly outperform or match strong baselines like AMPI-Q, TRPO, DQN, DDPG, TD3 and PPO. These results validate that RPI provides a robust and reliable foundation for building actor-critic algorithms that are less sensitive to the underlying model architecture.**

## I. INTRODUCTION

Function approximation (FA) is essential in reinforcement learning (RL) for tackling high-dimensional or continuous state-action spaces. However, its use fundamentally challenges the stability of classical Policy Iteration (PI), a core paradigm underlying many modern actor-critic algorithms such as PPO [1], DDPG [2] and TD3 [3]. These methods rely on alternating between policy evaluation (critic) and policy improvement (actor), closely reflecting the structure of PI. In the tabular setting, PI enjoys strong theoretical guarantees, most notably, monotonic improvement in value estimates with each iteration. But under general FA, these guarantees break down: inaccurate critic estimates can misguide policy updates, leading to suboptimal or even divergent behavior.

Reliable Policy Iteration (RPI) [4] is a recently proposed framework that permits greedy policy updates while provably restoring monotonic improvement in value estimates, even under general function approximation. RPI reformulates the policy evaluation step as a constrained optimization problem, ensuring that the estimated value function remains a pointwise lower bound of the true value function at every iteration. This constraint structure enables greedy policy updates while restoring the monotonic improvement guarantees that standard PI enjoys in the tabular setting, but under general FA.

The objective of this paper is to rigorously evaluate RPI's performance across a spectrum of function approximation classes. By systematically varying the architecture and capacity of the function approximator, we aim to assess whether RPI's theoretical benefits translate into empirical stability and robustness under different FA regimes.

Our key contributions are:

1) **Comprehensive Empirical Validation:** We conduct an empirical study of RPI, testing its performance and resilience across diverse benchmarks and function approximator capacities.
2) **Comparative Analysis:** We benchmark RPI against strong baselines (including DQN, PPO, DDPG, and TD3), providing clear evidence that it is resilient to the choice of function approximator class and consistently outperforms or matches the baselines.

The remainder of this paper is organized as follows. Section II briefly reviews the Reliable Policy Iteration (RPI) framework. Section III presents our comprehensive empirical evaluation, where we test RPI's resilience using both linear and deep neural network approximators across three benchmarks. Finally, Section IV concludes with a summary of our findings.

## II. RPI ALGORITHM DETAILS

In this section, we briefly review the Reliable Policy Iteration (RPI) framework, which was introduced in [4]. RPI is a variant of Policy Iteration (PI) designed to guarantee monotonic improvement and convergence under general function approximation, a setting where traditional PI methods often fail. We first describe the model-based version of the algorithm and then present its model-free critic loss, which can be readily integrated into modern deep reinforcement learning agents. For detailed theoretical analyses, proofs, and the derivation of the model-free loss, we refer the reader to the original work [4].

### A. Setup and Problem Formulation

We consider a stationary Markov Decision Process (MDP) denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent

[1]S.R. Eshwar and Gugan Thoppe are with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. {eshwarsr,gthoppe}@iisc.ac.in

[2]Kintan Saha and Krishna Agarwal are with the UG Department, Indian Institute of Science, Bangalore, India. {kintansaha,krishnaagarw}@iisc.ac.in

[3]Aniruddha Mukherjee is with the School of Computer Engineering, Kalinga Institute of Industrial Technology, Bhubaneswar, India. aniruddha.mukherjee@fsid-iisc.in

[4]Aditya Gopalan is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India.aditya@iisc.ac.in

[5]Gal Dalal is a Senior Research Scientist at NVIDIA Research. gdalal@nvidia.com

finite sets of states and actions, respectively, with cardinalities $|\mathcal{S}| = S$ and $|\mathcal{A}| = A$. The transition dynamics are governed by the kernel $\mathcal{P}$, where $\mathcal{P}(s' \mid s, a)$ indicates the probability of transitioning to state $s'$ upon taking action $a$ in state $s$. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a scalar reward to each state-action pair, and the discount factor $\gamma \in [0, 1)$ controls the trade-off between immediate and future rewards.

Let $\Delta(\mathcal{U})$ denote the set of probability distributions over a finite set $\mathcal{U}$. A stationary policy $\mu : \mathcal{S} \to \Delta(\mathcal{A})$ maps each state to a distribution over actions. The action-value function (or Q-function) associated with a policy $\mu$, denoted by $Q_\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, is defined as $Q_\mu(s, a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t), |, s_0 = s, , a_0 = a\right]$, where the dynamics evolve according to $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$ and $a_{t+1} \sim \mu(\cdot \mid s_{t+1})$ for $t \geq 0$.

The Bellman operator corresponding to a fixed policy $\mu$ is a mapping $T_\mu : \mathbb{R}^{SA} \to \mathbb{R}^{SA}$ defined as

$$(T_\mu Q)(s, a) := r(s, a) + \gamma \sum_{s', a'} \mathcal{P}(s' \mid s, a)\mu(a' \mid s')Q(s', a'). \tag{1}$$

Our goal is to construct a policy iteration (PI) algorithm suitable for use with function approximation (FA), while retaining the desirable properties of monotonic improvement and convergence characteristic of exact asynchronous PI. We denote the function approximation space by $\mathcal{F} \subseteq \mathbb{R}^{SA}$, where each function $f \in \mathcal{F}$ can be viewed as a real-valued mapping from $\mathcal{S} \times \mathcal{A}$ to $\mathbb{R}$, or equivalently, as a vector in $\mathbb{R}^{SA}$.

### *B. Model-Based RPI Algorithm*

The core novelty of RPI lies in its policy evaluation step. Unlike standard approaches that minimize Bellman or projection errors, RPI formulates a constrained optimization problem. This yields a value function estimate that serves as a guaranteed lower bound on the true value function and ensures monotonic improvement over iterations. This ensures that the subsequent greedy policy update is based on a more reliable signal.

Let $\mathcal{F}$ be a class of functions used to approximate the Q-value function. Starting with an initial policy $\mu_0$ and its corresponding Q-function approximation $f_0 \in \mathcal{F}$, RPI iteratively performs the two steps outlined in Algorithm 1.

The constraints in the policy evaluation step (Eq. 2) ensure that the sequence of value estimates $(f_k)$ is non-decreasing, and that each $f_k$ is a pointwise lower bound on the true Q-value function $Q_{\mu_k}$ [4, Theorem 3.1]. Together, these properties extend the monotonic improvement guarantees of tabular PI to the function approximation setting.

### *C. Model-Free RPI Critic Loss*

The model-based formulation in Algorithm 1 is impractical for many real-world applications, as it requires knowledge of the transition dynamics (to compute the Bellman operator $T_{\mu_k}$) and involves solving a constrained optimization over the entire state-action space. To overcome these limitations, a practical, model-free critic loss was derived

**Algorithm 1** Reliable Policy Iteration (RPI)

**Input:** FA class $\mathcal{F}$, policy $\mu_0$, an initial approximation $f_0 \in \mathcal{F}$ of $Q_{\mu_0}$, and a norm $\|\cdot\|$

**for** $k = 0, 1, 2$ ... until convergence **do**

**Policy Evaluation**:

$$\begin{aligned} f_{k+1} \in \arg\max_{f \in \mathcal{F}} \quad & \|f - f_k\| \\ \text{s.t.} \quad & T_{\mu_k} f \geq f \geq f_k. \end{aligned} \tag{2}$$

**Policy Improvement**:

$$\mu_{k+1} \in \{\mu : \mu \text{ is a deterministic policy that is greedy w.r.t. } f_{k+1}\} \tag{3}$$

**end for**

by reformulating the constrained optimization problem as an unconstrained one and using sample-based approximations [4]. This results in a unified objective that can replace the standard Mean Squared Bellman Error (MSBE) loss in actor-critic algorithms like Deep Q-Network (DQN) [5] and Deep Deterministic Policy Gradient (DDPG) [2]. The model-free RPI critic is trained by minimizing the following loss function

$$\mathcal{L}^{\text{RPI}}(f) := \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \Bigg[ -c \cdot f(s_i, a_i) + \lambda_1 \Big[ f(s_i, a_i) - y_i \Big]_+ + \lambda_2 \Big[ q_{\min} - f(s_i, a_i) \Big]_+ \Bigg]. \tag{4}$$

Here, $\mathcal{B}$ is a minibatch of transitions, $f$ is the critic network, $[x]_+ = \max(x, 0)$ is the ReLU function, and $c, \lambda_1, \lambda_2 > 0$ are hyperparameters of which $\lambda_1$ and $\lambda_2$ penalize constraint violations. The target value $y_i$ is computed analogously to standard DQN or DDPG, using a target network to ensure stability. The term $q_{\min}$ is a fixed, predefined lower bound on the Q-values, ensuring the estimates improve relative to a conservative baseline. This model-free critic loss encourages the critic to produce value estimates that are reliable lower bounds on the true discounted returns, thus mitigating the overestimation bias common in value-based methods. For a complete derivation of this model-free critic loss, see [4].

## III. EXPERIMENTS

In this section, we empirically validate the central claim of our work: that RPI demonstrates resilience and robust performance across a variety of function approximation classes. We have designed our experiments to showcase that RPI's theoretical guarantees of monotonic improvement and convergence translate into superior practical performance, regardless of the approximator's capacity or architecture. To provide a comprehensive evaluation, we assess both final policy performance and learning efficiency. For the latter, we measure the Area Under the Curve (AUC) of the learning curve, which reflects the cumulative rewards obtained during

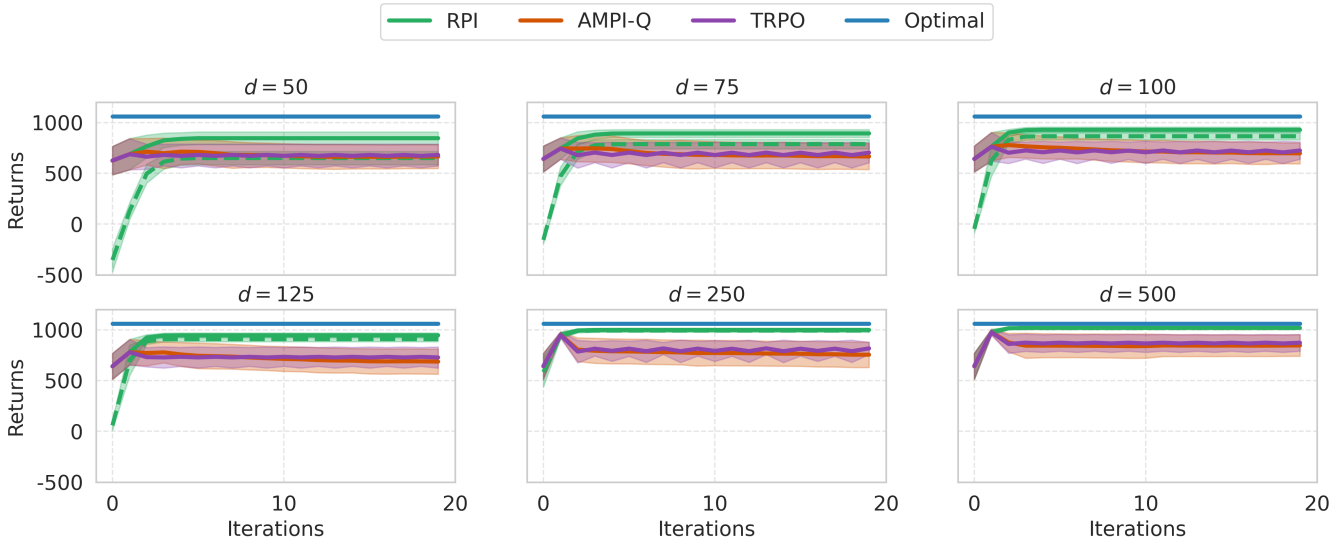


Fig. 1: Performance on Inventory Control with varying $d$. The curves show the mean return over 100 runs ($\pm$ standard deviation, shaded). Solid lines indicate the true policy return, and dashed lines represent the critic's estimated value. RPI not only outperforms AMPI-Q and TRPO but also consistently maintains its value estimates as a lower bound. As $d$ increases, RPI's approximation gap narrows and its final policy performance approaches the optimal value.

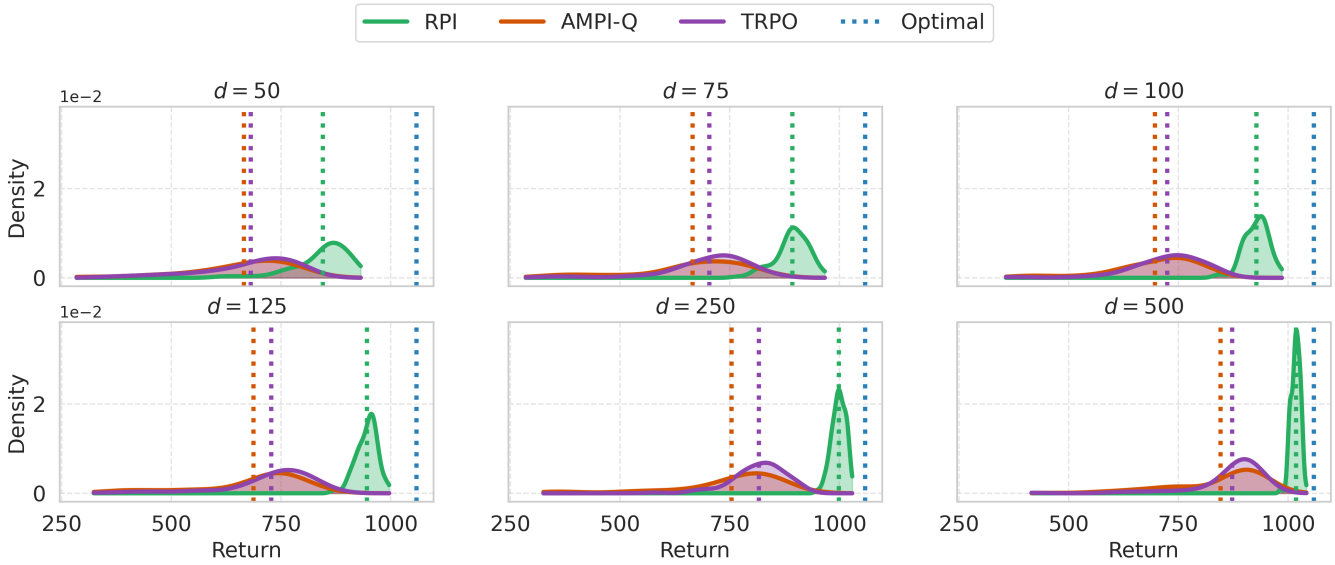


Fig. 2: Kernel Density Estimate (KDE) plots of the terminal policy returns over 100 runs, with dotted lines marking the mean for each algorithm. Across all dimensions ($d$), RPI shows a superior mean return and lower variance compared to AMPI-Q and TRPO. As $d$ increases, RPI's final policy performance approaches the optimal value while reducing the variance.

TABLE I: Comparison of RPI, AMPI-Q, and TRPO on the Inventory Control problem across varying function approximator dimensions ($d$) over 100 runs. Metrics are reported as mean $\pm$ standard deviation. RPI consistently outperforms the baselines in both terminal return and AUC, and its performance improves significantly as $d$ increases.

| Algorithm | Metric | d=50 | d=75 | d=100 | d=125 | d=250 | d=500 |
|---|---|---|---|---|---|---|---|
| **RPI** | Terminal Return | **845.2 ± 62.3** | **892.2 ± 38.2** | **927.7 ± 28.0** | **945.5 ± 23.7** | **998.3 ± 15.7** | **1017.5 ± 10.1** |
| | AUC ($\times 10^3$) | **15.7 ± 1.1** | **16.6 ± 0.7** | **17.3 ± 0.5** | **17.6 ± 0.47** | **18.7 ± 0.3** | **19.10 ± 0.2** |
| AMPI-Q | Terminal Return | 664.9 ± 117.4 | 665.3 ± 128.3 | 696.4 ± 103.1 | 686.5 ± 123.7 | 754.1 ± 125.5 | 845.4 ± 105.8 |
| | AUC ($\times 10^3$) | 12.9 ± 1.6 | 13.2 ± 1.9 | 13.8 ± 1.7 | 13.7 ± 1.8 | 14.8 ± 2.0 | 16.0 ± 1.8 |
| TRPO | Terminal Return | 680.6 ± 107.3 | 702.8 ± 98.0 | 724.4 ± 85.5 | 727.4 ± 104.0 | 816.5 ± 59.8 | 872.3 ± 82.5 |
| | AUC ($\times 10^3$) | 12.8 ± 1.9 | 13.2 ± 1.8 | 13.6 ± 1.6 | 13.9 ± 1.6 | 15.3 ± 1.4 | 16.5 ± 1.5 |

training. A higher AUC indicates better sample complexity, as an efficient agent quickly reaches a high-performing policy and maintains it, maximizing the total discounted return gathered over the training period.

Our evaluation is structured into three parts. First, we investigate a model-based setting using linear function approximation on an Inventory Control problem, where we vary the feature dimension $d$. Second and third, we move to model-free deep reinforcement learning settings, evaluating RPI-enhanced agents $\text{RPI}_{\text{DQN}}$ and $\text{RPI}_{\text{DDPG}}$ on the classic CartPole-v1 [6] (discrete actions) and InvertedPendulum-v5 [6], [7] (continuous actions) benchmarks respectively, where we modify the neural network architecture that parameterizes the Q-value network. Across all settings, RPI consistently matches or surpasses the performance of strong baselines on all metrics.

### A. Inventory Control

We first assess RPI in a model-based setting with linear function approximation on the standard inventory control problem, following the exact same environmental setup and parameters as in [4]: maximum inventory capacity $M = 49$ (hence, $|\mathcal{S}| = |\mathcal{A}| = 50$), unit cost $c = 5$, holding cost $h = 1$, and selling price $p = 10$. We presume the daily demand follows a uniform distribution over $\{0, \dots, M\}$ and the discount factor $\gamma = 0.9$. We consider a linear FA setting for this problem, i.e., we let $\mathcal{F} = \{\Phi\theta : \theta \in \mathbb{R}^d\}$, where $\Phi \in \mathbb{R}^{SA \times d}$ is the feature matrix and $d \ll |\mathcal{S}||\mathcal{A}|$ is $\mathcal{F}$'s dimension. In all our runs, the entries of $\Phi$, i.e., the features, are sampled uniformly from the interval $[1, 5]$. We compare RPI against two representative baselines: AMPI-Q [8], an algorithm using a fitted critic, and TRPO [9], a state-of-the-art method with a cautious actor.

To test the resilience of each algorithm to the capacity of the function approximator, we vary the dimensionality of the feature space, $d$. We conduct experiments for $d \in \{50, 75, 100, 125, 250, 500\}$. For each value of $d$, we run 100 independent trials and compare the algorithms based on their learning stability and final policy performance.

The learning dynamics, illustrated in Figure 1, highlight RPI's unique stability. Solid lines indicate the true policy value, $\mathbb{E}_{\nu,\mu_k}[Q_{\mu_k}(s,a)]$, while dashed lines represent the estimated value, $\mathbb{E}_{\nu,\mu_k}[\phi^\top(s,a)\theta_k]$, where $\nu$ is the uniform distribution over states. The learning curves are averaged over 100 independent seeds, with the shaded band denoting one standard deviation around the mean. RPI's estimates are monotonically non-decreasing and consistently lie below the true values, providing an explicit lower bound throughout training. As the function approximator's capacity increases (i.e., larger $d$), the gap between RPI's estimated and true values diminishes, indicating improved accuracy.

This resilience translates into superior performance, as quantified in Table I. Across all $d$ values, RPI significantly outperforms both AMPI-Q and TRPO in both mean terminal return and AUC. The distribution of these terminal returns, visualized in the KDE (Kernel Density Estimation) plots in Figure 2, offers further insight. As $d$ increases, RPI's mean performance not only improves but its variance also markedly decreases, resulting in a tightening distribution around a near-optimal value. This combined evidence showcases RPI's robustness; its ability to find high-performing and reliable policies is not hindered by the choice of function approximator capacity.

### B. Cartpole

Next we evaluate the RPI critic (Eq. 4) in a model-free setting on the classic CartPole-v1 benchmark. To assess the resilience of our $\text{RPI}_{\text{DQN}}$ variant to network capacity, we compare it against DQN and PPO using standard implementations from the Stable-Baselines3 library [10] and finetuned hyperparameters from Stable-Baselines3 RL Zoo. For this experiment, we replace only the default value estimation network in Stable-Baselines3 — a two-layer neural network — with a single layer network while all other hyperparameters remain unchanged from those in [4]. We then vary the number of neurons in this single layer across the set {8, 16, 32, 64, 128, 256}, conducting 25 independent runs for each architectural configuration.

The results highlight $\text{RPI}_{\text{DQN}}$'s robustness to variations in network capacity. Figure 3 presents the learning dynamics averaged over 25 independent seeds, with the shaded band denoting one standard deviation. The individual lines within the plots distinguish the policy's true discounted return (solid curves), obtained by averaging 15 Monte-Carlo roll-outs, from the critic's predicted Q-values for the initial state–action pairs of those roll-outs (dotted curves). These curves show that $\text{RPI}_{\text{DQN}}$ demonstrates effective learning even with low-capacity networks (8 and 16 neurons), a regime where DQN struggles. For architectures with 32 or more neurons, $\text{RPI}_{\text{DQN}}$ consistently achieves faster convergence and superior performance compared to both DQN and PPO baselines.

These qualitative findings are substantiated by the aggregate metrics in Table II. The data reveals that $\text{RPI}_{\text{DQN}}$ achieves higher sample efficiency (AUC) than both baselines for networks of 16 or more neurons, evidenced by a higher mean and significantly lower variance. In terms of final policy performance, $\text{RPI}_{\text{DQN}}$ consistently surpasses DQN and matches the performance of PPO for most tested architectures (16+ neurons), confirming its ability to produce stable, high-performing policies across a wide range of model sizes.

### C. Inverted Pendulum

Next, we evaluate $\text{RPI}_{\text{DDPG}}$ against PPO, DDPG, and TD3 on the InvertedPendulum-v5 benchmark. For this experiment, we vary the architecture of the two-layer actor and critic networks across several configurations ({32-32, 64-64, 128-128, 256-256, 512-512}, as well as the default 400-300), while all other hyperparameters remain identical to those in [4].

The results highlight $\text{RPI}_{\text{DDPG}}$'s robustness to variations in network capacity. Figure 4 presents the learning dynamics averaged over 25 independent seeds, with the shaded band denoting one standard deviation. The plots distinguish the policy's true discounted return (solid curves), obtained

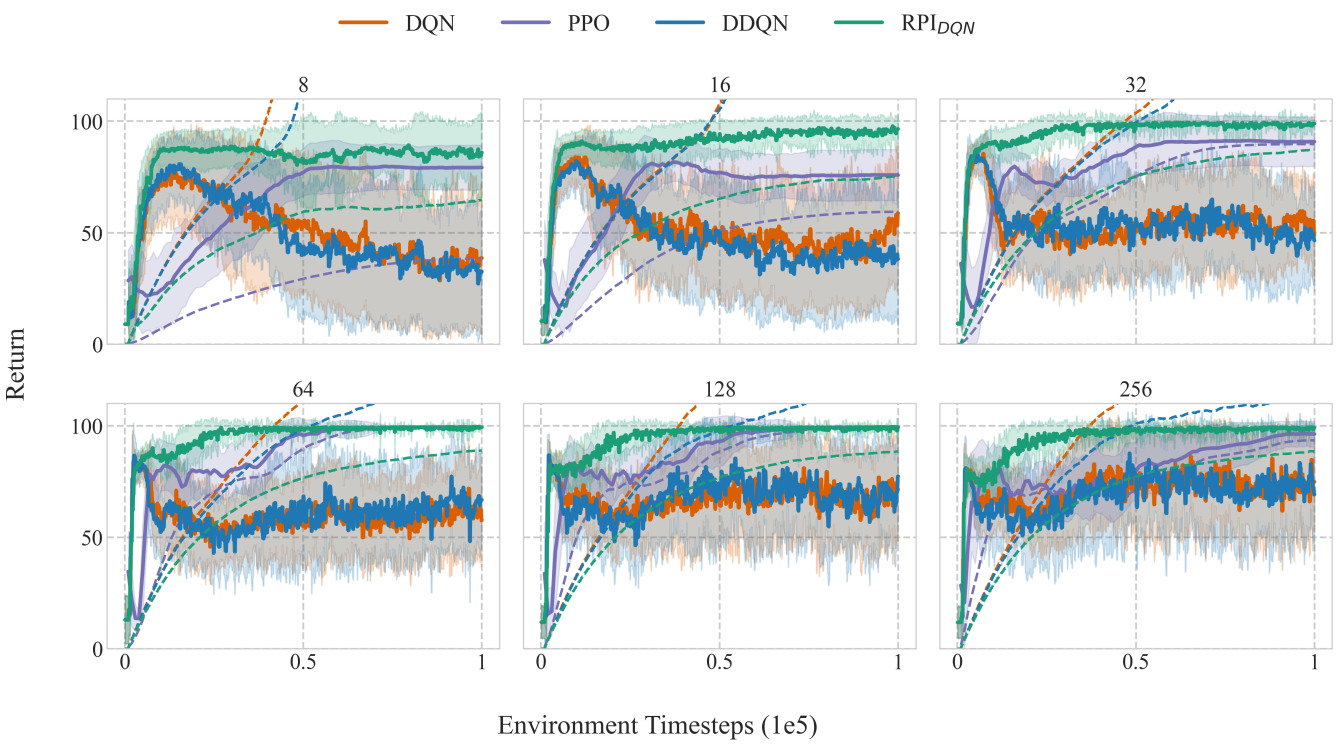


Fig. 3: Training Curves for Cartpole-v1 using different neural network sizes. (solid: true discounted Monte Carlo return, dashed: estimated value function). Cumulative performance over 25 runs. $RPI_{DQN}$ performance is comparable to PPO, while maintaining lower bound property and has higher AUC.

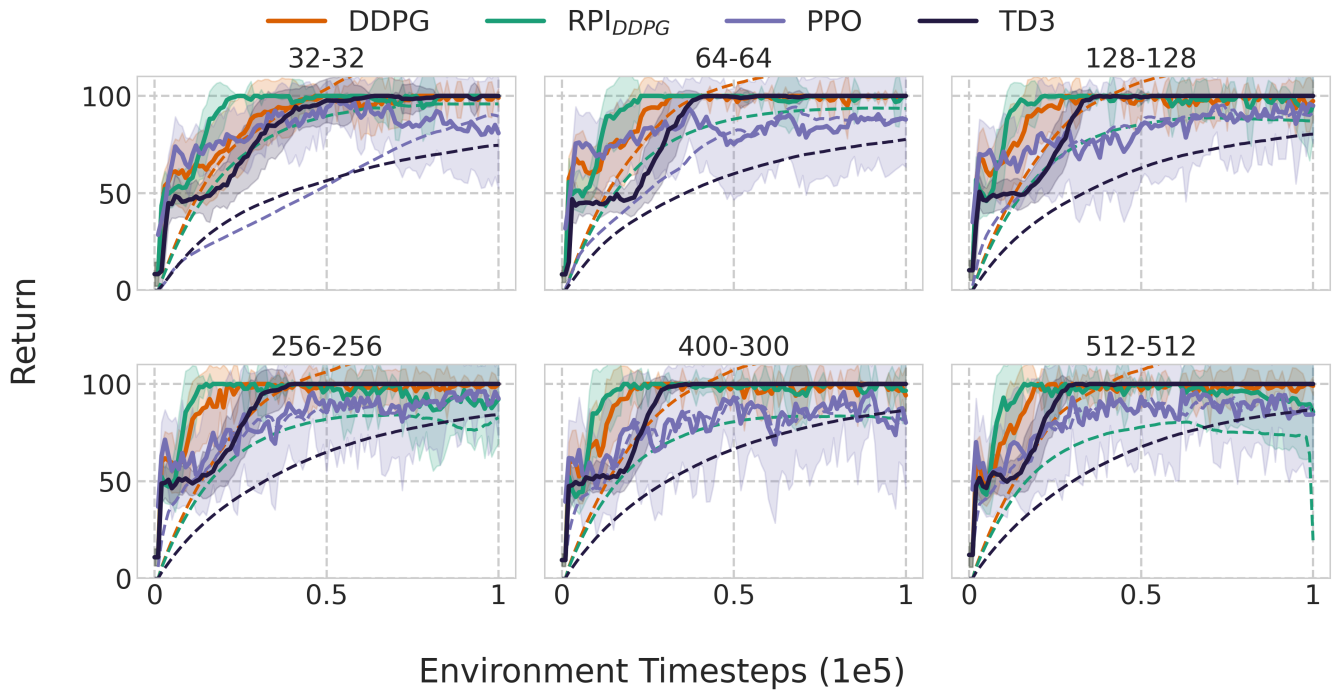


Fig. 4: Training Curves for Inverted Pendulum-v5 using different neural network sizes. (solid: true discounted Monte Carlo return, dashed: estimated value function). Cumulative performance over 25 runs. $\text{RPI}_{\text{DDPG}}$ is the best performing algorithm while maintaining the lower bound property and having higher AUC.

TABLE II: Quantitative comparison of $\text{RPI}_{DQN}$ against DQN and PPO baselines on CartPole across **single-layer** network architectures. We report the mean ± standard deviation over 25 random seeds. **Final Performance** is the average return over the final 10% of training steps. **AUC** measures overall sample efficiency.

| Algorithm | Metric | Net-Arch | | | | | |
|---|---|---|---|---|---|---|---|
| | | **8** | **16** | **32** | **64** | **128** | **256** |
| **DQN** | **Final Perf.** | 24.9 ± 29.69 | 46.3 ± 31.9 | 65.1 ± 28.9 | 67.3 ± 19.1 | 71.7 ± 16.9 | 85.6 ± 11.8 |
| | **AUC (x10$^6$)** | 2.8 ± 2.3 | 5.6 ± 2.1 | 6.6 ± 1.6 | 7.3 ± 0.6 | 7.3 ± 0.8 | 7.8 ± 0.4 |
| **PPO** | **Final Perf.** | **82.9 ± 14.0** | **94.5 ± 8.6** | **99.1 ± 1.2** | **99.3 ± 0.0** | **99.3 ± 0.0** | 98.9 ± 2.2 |
| | **AUC (x10$^6$)** | **6.5 ± 0.6** | 7.3 ± 0.7 | 8.3 ± 0.5 | 8.7 ± 0.3 | 8.8 ± 0.3 | 8.2 ± 0.5 |
| $\text{RPI}_{DQN}$ | **Final Perf.** | 54.3 ± 31.7 | 87.6 ± 13.1 | 97.8 ± 2.5 | 99.3 ± 0.2 | 99.3 ± 0.1 | **99.3 ± 0.1** |
| | **AUC (x10$^6$)** | 5.1 ± 2.3 | **8.3 ± 0.7** | **9.2 ± 0.3** | **9.3 ± 0.2** | **9.4 ± 0.1** | **9.4 ± 0.0** |

TABLE III: new Quantitative comparison of $\text{RPI}_{DQN}$ against DQN and PPO baselines on CartPole across **single-layer** network architectures. We report the mean ± standard deviation over 25 random seeds. **Final Performance** is the average return over the final 10% of training steps. **AUC** measures overall sample efficiency.

| Algorithm | Metric | Network Size | | | | | |
|---|---|---|---|---|---|---|---|
| | | **8** | **16** | **32** | **64** | **128** | **256** |
| **DQN** | **Final Perf.** | 37.89 ± 28.12 | 48.29 ± 22.85 | 54.95 ± 11.02 | 61.91 ± 12.13 | 68.25 ± 9.74 | 73.44 ± 8.96 |
| | **AUC (x10$^6$)** | 5.09 ± 1.93 | 5.22 ± 1.08 | 5.38 ± 0.57 | 5.90 ± 0.59 | 6.65 ± 0.51 | 7.06 ± 0.39 |
| **DDQN** | **Final Perf.** | 33.66 ± 24.69 | 40.22 ± 23.19 | 47.71 ± 14.61 | 65.82 ± 12.82 | 67.61 ± 9.02 | 73.53 ± 8.26 |
| | **AUC (x10$^6$)** | 4.96 ± 1.55 | 4.87 ± 1.13 | 5.40 ± 0.87 | 6.00 ± 0.57 | 6.72 ± 0.36 | 6.85 ± 0.29 |
| **PPO** | **Final Perf.** | 79.18 ± 10.05 | 75.87 ± 11.38 | 90.81 ± 11.13 | **99.34 ± 0.00** | **99.34 ± 0.00** | 96.27 ± 5.68 |
| | **AUC (x10$^6$)** | 6.36 ± 0.75 | 6.72 ± 0.79 | 7.89 ± 0.79 | 8.70 ± 0.24 | 8.70 ± 0.33 | 7.99 ± 0.62 |
| $\text{RPI}_{\text{DQN}}$ | **Final Perf.** | **85.65 ± 11.33** | **95.24 ± 4.35** | **98.22 ± 2.28** | 98.84 ± 0.77 | 98.80 ± 0.41 | **98.79 ± 0.34** |
| | **AUC (x10$^6$)** | **8.27 ± 0.83** | **8.98 ± 0.40** | **9.44 ± 0.13** | **9.46 ± 0.06** | **9.39 ± 0.08** | **9.33 ± 0.07** |

TABLE IV: Quantitative comparison of $\text{RPI}_{DQN}$ against PPO, DDPG and TD3 baselines on Inverted Pendulum across **double-layer** network architectures. We report the mean ± standard deviation over 25 random seeds. **Final Performance** is the average return over the final 10% of training steps. **AUC** measures overall sample efficiency.

| Algorithm | Metric | Net-Arch | | | | | |
|---|---|---|---|---|---|---|---|
| | | **32-32** | **64-64** | **128-128** | **256-256** | **400-300** | **512-512** |
| **DDPG** | **Final Perf.** | 99.3 ± 2.8 | 99.7 ± 0.8 | 97.0 ± 6.0 | 99.2 ± 1.6 | 98.3 ± 2.9 | 98.6 ± 2.6 |
| | **AUC (x10$^6$)** | 8.8 ± 0.3 | 9.1 ± 0.2 | 9.2 ± 0.2 | **9.3 ± 0.3** | 9.2 ± 0.2 | **9.2 ± 0.2** |
| **PPO** | **Final Perf.** | 82.7 ± 21.5 | 87.6 ± 23.8 | 92.8 ± 14.1 | 94.1 ± 13.3 | 81.8 ± 23.0 | 86.0 ± 23.0 |
| | **AUC (x10$^6$)** | 8.2 ± 0.6 | 8.1 ± 1.0 | 8.0 ± 1.1 | 8.1 ± 1.2 | 7.8 ± 0.9 | 8.1 ± 0.8 |
| **TD3** | **Final Perf.** | 99.6 ± 2.08 | **100.0 ± 0.0** | **100.0 ± 0.0** | **100.0 ± 0.0** | **100.0 ± 0.0** | **100.0 ± 0.0** |
| | **AUC (x10$^6$)** | 8.3 ± 0.4 | 8.3 ± 0.3 | 8.6 ± 0.2 | 8.7 ± 0.2 | 8.8 ± 0.2 | 8.9 ± 0.2 |
| $\text{RPI}_{DDPG}$ | **Final Perf.** | **100.0 ± 0.1** | 99.3 ± 2.6 | 95.9 ± 9.3 | 90.3 ± 17.1 | 98.4 ± 5.9 | 90.4 ± 25.5 |
| | **AUC (x10$^6$)** | **9.2 ± 0.2** | **9.3 ± 0.2** | **9.3 ± 0.1** | 9.2 ± 0.4 | **9.3 ± 0.2** | **9.2 ± 0.7** |

by averaging 100 Monte-Carlo roll-outs, from the critic's predicted Q-values for the initial state–action pairs (dotted curves). These curves show that across all tested architectures, including low-capacity networks, $\text{RPI}_{\text{DDPG}}$ consistently achieves faster convergence compared to the PPO, DDPG, and TD3.

These qualitative findings are substantiated by the aggregate metrics in Table IV. The data reveals that $\text{RPI}_{\text{DDPG}}$ matches or exceeds the sample efficiency (AUC) of all baselines. In terms of final policy performance, $\text{RPI}_{\text{DDPG}}$ consistently matches the top-performing baseline for most tested architectures, confirming its ability to produce stable, high-performing policies across a wide range of model sizes.

## IV. CONCLUSIONS

In conclusion, this work provides evidence that RPI is highly robust to the choice of function approximator. Across experiments on Inventory Control, CartPole-v1, and InvertedPendulum-v5, RPI consistently matched or outperformed state-of-the-art baselines in both final policy performance and sample efficiency. This advantage was particularly pronounced when using low-capacity function approximators (small d values and simpler network architectures), confirming RPI's stability. A natural direction for future work is to extend this evaluation to more complex environments, such as the Atari and MuJoCo suites, to further demonstrate RPI's performance across a wide range of function approximation classes.

## REFERENCES


[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[3] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[4] S. R. Eshwar, G. Thoppe, A. Gopalan, and G. Dalal, "Reliable critics: Monotonic improvement and convergence guarantees for reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2506.07134

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[7] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[8] B. Scherrer, M. Ghavamzadeh, V. Gabillon, B. Lesner, and M. Geist, "Approximate modified policy iteration and its application to the game of tetris," *Journal of Machine Learning Research*, vol. 16, no. 49, pp. 1629–1676, 2015. [Online]. Available: http://jmlr.org/papers/v16/scherrer15a.html

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[10] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html