RELATIVE ENTROPY PATHWISE POLICY OPTIMIZATION

Claas A Voelcker
University of Toronto
Vector Institute
Toronto, Canada

Axel Brunnbauer
Technische Universität Wien
Vienna, Austria

Marcel Hussing
University of Pennsylvania
Philadelphia, USA

Michal Nauman
University of Warsaw
UC Berkeley
Warsaw, Poland

Pieter Abbeel
UC Berkeley
Berkeley, USA

Eric Eaton
University of Pennsylvania
Philadelphia, USA

Radu Grosu
Technische Universität Wien
Vienna, Austria

Amir-massoud Farahmand
Polytechnique Montréal
Mila – Quebec AI Institute
University of Toronto
Montreal, Canada

Igor Gilitschenski
University of Toronto
Vector Institute
Toronto, Canada

arXiv:2507.11019v2 [cs.LG] 18 Jul 2025

ABSTRACT

Score-function policy gradients (Sutton et al., 1999) have delivered strong results in game-playing, robotics and language-model fine-tuning. Yet its high-variance often undermines training stability. On the other hand, pathwise policy gradients (Silver et al., 2014) alleviate the training variance, but are reliable only when driven by an accurate action-conditioned value function which is notoriously hard to train without relying on past off-policy data. In this paper, we discuss how to construct a value-gradient driven, on-policy algorithm that allow training Q-value models purely from on-policy data, unlocking the possibility of using pathwise policy updates in the context of on-policy learning. We show how to balance stochastic policies for exploration with constrained policy updates for stable training, and evaluate important architectural components that facilitate accurate value function learning. Building on these insights, we propose *Relative Entropy Pathwise Policy Optimization (REPPO)*, an efficient on-policy algorithm that combines the sample-efficiency of pathwise policy gradients with the simplicity and minimal memory footprint of standard on-policy learning. We demonstrate that REPPO provides strong empirical performance at decreased sample requirements, wall-clock time, memory footprint as well as high hyperparameter robustness in a set of experiments on two standard GPU-parallelized benchmarks.

1 INTRODUCTION

Score-function policy gradient methods such as Reinforce (Williams, 1992) or PPO (Schulman et al., 2017) provide a principled way for data-driven optimization of the parameterized policies and have proved highly effective for robotic control (Rudin et al., 2022; Kaufmann et al., 2023; Radosavovic et al., 2024), and language-model fine-tuning (Ouyang et al., 2022; Touvron et al., 2023; Gao et al., 2023; Liu et al., 2024). Justified by the *Policy Gradient Theorem* (Sutton et al., 1999), these methods rely on a Monte-Carlo approximation of the gradient of log-probabilities of the sampled actions, without directly backpropagating through the return function. Because this gradient carries no information about how performance varies under small action perturbations (Heess et al., 2015), it exhibits high variance (Greensmith et al., 2004) and often leads to unstable learning (Ilyas et al., 2020; Rahn et al., 2023), especially in high-dimensional continuous spaces (Li et al., 2018).

A popular alternative to score-based estimators is the pathwise policy gradient (Silver et al., 2014), in which the gradient is obtained via backpropagating through a learned surrogate model of the return called a value function. This provides a low-variance signal that directly optimizes the predicted returns and often leads to faster learning than the score-based alternative (Lillicrap et al., 2016).

However, the effectiveness of pathwise policy gradients is restricted by the quality of the approximate value function (Silver et al., 2014). As such, algorithms that use pathwise policy gradients usually rely on stabilizing the value learning through off-policy training (Fujimoto et al., 2018; Haarnoja et al., 2018). Unfortunately, off-policy training requires the use of large replay buffers. The first obvious drawback is that simply storing the replay buffer can create a challenge when the number of samples exceeds the memory availability. In addition, training with past policy data introduces various well-studied challenges for value function fitting (Thrun & Schwartz, 1993; Baird, 1995; Van Hasselt, 2010; Sutton et al., 2016; Kumar et al., 2021; Nikishin et al., 2022; Lyle et al., 2024; Hussing et al., 2024; Voelcker et al., 2025). This raises the following fundamental question:

Can we train a robust surrogate value function and effectively use pathwise policy gradient in a fully on-policy setting?

Leaning on the progress in accurate value function learning (Sutton, 1988; Haarnoja et al., 2019; Schwarzer et al., 2021; Hussing et al., 2024; Farebrother et al., 2024), this paper can answer the above affirmatively. We propose an efficient on-policy algorithm, *Relative Entropy Pathwise Policy Optimization (REPPO)*, which uses the pathwise policy gradient estimator driven by an accurate surrogate value model learned from on-policy data. REPPO builds on the maximum entropy framework (Ziebart et al., 2008) to encourage continual exploration. It combines this with a principled KL regularization scheme inspired by the Relative Entropy Policy Search method (Peters et al., 2010) to prevent aggressive policy updates from destabilizing the optimization. Furthermore, we incorporate several recent advances in neural network architecture design to stabilize learning: categorical Q-learning, appropriately normalized neural network architectures, and auxiliary tasks.

We test our approach in a variety of locomotion and manipulation environments from Mujoco Playground (Zakka et al., 2025) and ManiSkill (Tao et al., 2025) benchmarks, and show that REPPO is able to achieve superior performance and wall-clock efficiency when compared to tuned on-policy baselines, while using significantly smaller memory footprints than comparable off-policy algorithms. Furthermore, we find that the proposed method is robust to the choice of hyperparameters. To this end, our method offers stable performance across more than 30 tasks spanning multiple benchmarks with a single hyperparameter set. Our contributions are as follows:

1. We present REPPO which showcases the effectiveness of the pathwise policy gradient in on-policy RL with continuous action spaces.
2. We show how a joint entropy and policy deviation tuning objective can address the twin problems of sufficient exploration and controlled policy updates.
3. We evaluate architectural components such as cross-entropy losses, layer normalization, and auxiliary tasks for their efficacy in pathwise policy gradient-based on-policy learning.

To facilitate reproducibility, we provide sample implementations in both the jax (Bradbury et al., 2018) and torch (Paszke et al., 2019) frameworks. Our code is available under the following address: `https://github.com/cvoelcker/reppo`.

2 Background, Notation, and Definitions

We consider the setting of the Markov Decision Process (MDP) (Puterman, 1994) , defined by the tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, r, \gamma, \rho_0)$, where $\mathcal{X}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}(x'|x, a)$ is the transition probability kernel, $r(x, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor that trades off immediate and future rewards. We write $\mathcal{P}_\pi(x'|x)$ for the policy-conditioned transition kernel and $\mathcal{P}^n_\pi(y|x)$ for the n-step transition kernel. We use $\mu_\pi(y|x)$ to denote the discounted stationary distribution over states y when starting in state x. Formally

$$\mu_\pi(y|x) = \frac{1}{1-\gamma} \sum_{i=0}^{\infty} \gamma^i \mathcal{P}^i_\pi(y|x) \tag{1}$$

When $x \sim \mu_\pi(\cdot|y)$, $y \sim \rho_0$, and the policy is clear from context, we will simply write $\mu_\pi(x)$ to denote the probability of a state under the discounted occupancy distribution.

An agent interacts with the environment via a policy $\pi(a|x)$, which defines a distribution over actions given a state. The objective is to find a policy that maximizes the expected discounted return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \right], \tag{2}$$

where $x_0 \sim \rho_0$ is the initial state distribution, and $a_t \sim \pi(\cdot|x_t)$. The state-action value function associated with a policy π are defined as:

$$Q^\pi(x, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \Big| x_0 = x, a_0 = a \right]. \tag{3}$$

The optimal policy π^* maximizes the expected return from every state, with corresponding optimal value functions $Q^*(x, a)$.

2.1 Bootstrapped value learning

To train value function approximations, current deep RL methods, both on- and off-policy, mostly use a bootstrapped regression loss together with a parametric approximation of the Q function

$$\mathcal{L}_Q(\theta|x, a, x') = \left[Q_\theta(x, a) - G(x, a)_{\text{sg}} \right]^2, \tag{4}$$

where $G(x, a)$ is a target value estimate and $[]_{\text{sg}}$ denotes the stop-gradient operator. The simplest form of target estimate is the one step value bootstrap

$$G(x, a) = r(x, a) + \gamma Q_\theta(x', a'), \tag{5}$$

where a' is chosen according to a policy, either the current target or an estimate of the optimal policy.

However, it is also possible to bootstrap with multiple timesteps, leading to an n-step return estimate

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r(x_{t+k}, a_{t+k}) + \gamma^n V^\pi(x_{t+n}). \tag{6}$$

In this formulation, the sequence needs to be gathered form a fixed policy, otherwise importance sampling (IS) correction needs to be used to account for changes. Longer n-step returns tend to have higher variance, but rely less on the correctness of the tail value function estimate, which is exponentially discounted by γ^n.

TD(λ) (Sutton, 1988) is a Temporal-Difference learning method which unifies multi-step and single-step bootstraping through a trace-decay parameter $\lambda \in [0, 1]$. TD(λ) estimates a target value function $Q^\pi(x, a)$ by combining multi-step returns using exponentially weighted averaging. The n-step return from time t is defined as: and the λ-return is a weighted sum of all n-step returns:

$$G_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}. \tag{7}$$

TD(λ) balances the bias and variance tradeoff in the bootstrap estimates. Setting $\lambda = 0$ corresponds to one-step TD learning, while $\lambda = 1$ recovers the multi-step method .

2.2 Policy gradient learning

A policy gradient approach (Sutton & Barto, 2018) is a general method for improving a (parameterized) policy π_θ by estimating the gradient of the policy-return function $J(\pi_\theta)$ with regard to the policy parameters θ. The *policy gradient theorem* states that

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{x \sim \mu_\pi}[Q(x, a) \nabla_\theta \log \pi_\theta(a|x)]. \tag{8}$$

This identity is particularly useful as both the Q value and the stationary distribution can be estimated by samples obtained from following the policy for sufficiently many steps in the environment.

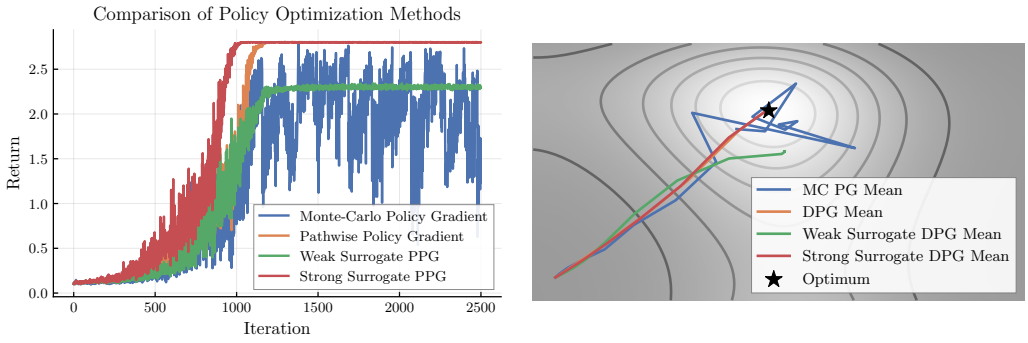


Figure 1: Achieved returns (left) and path of four policies trained with different gradient estimation methods. We compare a Monte-Carlo based policy gradient estimator (blue) with three variants of pathwise gradient estimators: using the ground truth objective function (orange), an inaccurate surrogate model (green), and an accurate surrogate model (red). All PPG based methods show markedly reduced variance in the policy updates.

The main drawback of this approach is that sample-based estimators of the policy gradient tend to have very high variance, necessitating large amounts of data and small step sizes for accurate estimation. In addition, the policy gradient requires on-policy updates, which prevents these methods from efficiently leveraging offline data. While this can be partially overcome by importance sampling, this again introduces variance to the method, which can lead to prohibitively unstable algorithms.

An alternative approach, mostly leveraged in off-policy learning, is the *deterministic policy gradient theorem* (DPG) (Silver et al., 2014). While the name stresses the *deterministic* nature of the policy for historical reasons, the method has also been used with stochastic policies which admit a reparameterization, such as parameterized Gaussian policies (Haarnoja et al., 2018). We will therefore refer to it also as the *pathwise policy gradient*, following the nomenclature of Mohamed et al. (2020). The gradient estimator for the DPG relies on access to a differentiable state-action value function. While access to the ground-truth state-action value function is clearly not a realistic assumption, we can use learned surrogate models such as neural networks to obtain a biased estimate of the gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{x\sim\mu_\pi}[\nabla_a Q(x,a)|_{a=\pi_\theta(x)} \nabla_\theta \pi_\theta(x)]. \tag{9}$$

Essentially, the pathwise policy gradient replaces the Monte-Carlo policy gradient with a pathwise derivative through a surrogate model.

2.3 Illustrating Gradient Estimation

To build additional intuition on the differences between different policy gradient estimators, we conduct an illustrative experiment.

On a simple objective $g(x)$ we initialize four Gaussians and update their parameters to maximize

$$J(\mu, \Sigma) = \mathbb{E}_{x\sim\mathcal{N}(\cdot|\mu,\Sigma)}[g(x)] \tag{10}$$

with four different methods: a Monte-Carlo policy gradient (using Equation 8), a pathwise policy gradient with the ground truth objective function, and two pathwise policy gradients using learned approximations, one accurate and one inaccurate (all using Equation 9). We visualize the returns and the path of the mean estimates in Figure 1. The experiments highlight Monte-Carlo Policy Gradients have high variance, and can lead to unstable policies which fail to optimize the target. Differentiating the ground-truth objective in reinforcement learning requires differentiable simulators, which have indeed been used to train policies (Xu et al., 2022). But assuming access to ground-truth gradients is unrealistic in many scenarios, and in non-smooth regimes such as contact-rich simulation, direct differentiation can lead to unstable gradients (Metz et al., 2021; Suh et al., 2022a).

Therefore the use of surrogate models is necessary and sometimes even advantageous for general-purpose RL. Our experiment further shows the importance of this function. An incorrect surrogate function leads to low variance updates, but the method converges to a wrong solution. With a well-fit surrogate function, the pathwise policy gradient can lead to significantly improved learning.

Algorithm 1: Pseudocode for Relative Entropy Pathwise Policy Optimization

Input: Environment $\mathcal{E}$, actor network π_θ, critic network Q_ϕ, hyperparameters
Output: Trained policy π_θ
// Initialize networks
Actor π_θ, behavior policy $\pi_{\theta'}$ with $\theta' = \theta$, critic Q_ϕ with encoder f_ϕ, entropy and KL temperature α and β
for *iteration* $= 1$ *to* $N_{iterations}$ **do**
  // Step 1: Collect rollout with behavior policy
  for *step* $= 1$ *to* N_{steps} **do**
    // Apply exploration noise scaling
    Sample action $a_t \sim \pi_{\theta'}(\cdot|x_t)$
    Execute a_t in environment, observe (x_{t+1}, r_t, d_t)
    Compute $V_{t+1} = Q(x_{t+1}, a_{t+1})$ with $a_{t+1} \sim \pi_{\theta'}(\cdot|x_{t+1})$
    Compute $\psi_t = f_\phi(x_{t+1}, a_{t+1})$
    // Maximum entropy augmented reward, see Subsubsection 3.1.2
    $\tilde{r}_t = r_t - \alpha \log \pi_\theta(a_{t+1}|x_{t+1})$
    Store transition $(x_t, a_t, \tilde{r}_t, x_{t+1}, d_t, V_{t+1}, \psi_t)$
  end
  // Step 2: Compute TD-λ targets, see Subsubsection 3.1.1
  for $t = T - 1$ *down to* 0 **do**
    $G_t^\lambda = \tilde{r}_t + \gamma[(1 - d_t)(\lambda G_{t+1}^\lambda + (1 - \lambda)V_{t+1})]$
  end
  // Step 3: Update networks for multiple epochs
  for *epoch* $= 1$ *to* N_{epochs} **do**
    Shuffle data and create mini-batches
    for *each mini-batch* $b = \{(x, a, G^\lambda, \psi)_i\}_{i=1}^B$ **do**
      // Categorical critic update, see Subsubsection 3.2.1
      $L_Q = \frac{1}{B} \sum \text{CrossEntropy}(Q_\phi(x_i, a_i), \text{Cat}(G_i^\lambda))$
      // Auxiliary task, see Subsubsection 3.2.3
      $L_{aux} = \frac{1}{B} \sum ||f_\phi(x_i, a_i) - \psi_i||^2]$
      Update critic: $\phi \leftarrow \phi - \alpha_Q \nabla_\phi (L_Q + \beta L_{aux})$
      // Actor update with entropy and KL regularization, see Subsubsection 3.1.2 and Subsubsection 3.1.3
      Sample actions $a_i' \sim \pi_\theta(\cdot|x_i)$
      Compute KL divergence: $D_{KL}(x_i) = \text{KL}(\pi_\theta(x_i)||\pi_{\theta'}(x_i))$
      Policy loss: $L_\pi = -\frac{1}{B} \sum Q_\phi(x_i, a_i') + e^\alpha \log \pi_\theta(a_i'|x_i) + e^\beta D_{KL}(x_i)$
      Update actor: $\theta \leftarrow \theta - \eta_\pi \nabla_\theta L_\pi$
      Entropy α update: $\alpha \leftarrow -\eta_\alpha \nabla_\alpha e^\alpha (\frac{1}{B} \sum \mathcal{H}[\pi_\theta(x_i)] - \mathcal{H}_{\text{target}})$
      KL β update: $\beta \leftarrow -\eta_\beta \nabla_\beta e^\beta (\frac{1}{B} \sum D_{KL}(x_i)] - \text{KL}_{\text{target}})$
    end
  end
  // Behavior Policy Update
  $\theta' \leftarrow \theta$
end
return *Trained policy* π_θ

To use pathwise gradients in on-policy learning, our goal is thus to learn a suitable value function that allows us to estimate a low variance update direction without converging to a suboptimal solution.

3 Relative Entropy Pathwise Policy Optimization

The goal of this section is to establish an algorithmic recipe that allows us to use the pathwise policy gradient in an on-policy setting. Naively, one could attempt to take an off-policy algorithm like SAC

and train it solely with data from the current policy. However, Seo et al. (2025) recently showed this can quickly lead to unstable learning. To succeed in the on-policy regime, we need to be able to continually obtain new diverse data, and compute stable and reliable updates. The main challenge for the latter is to obtain a critic estimate that is accurate enough so that we benefit from the reduced variance of the gradient of the return without introducing excessive bias. Lastly, as available data in the on-policy regime is limited, a carefully constrained amount of policy change per step is vital.

We find that combining a set of traditional and recent advances in both reinforcement learning as well as neural network value function fitting can satisfy all the requirements above. We first introduce the core RL algorithm, and then elaborate on the architectural design of the method. The whole algorithm is summarized in pseudo-code in Algorithm 1 with references to each section.

3.1 RL Algorithm

At its core, REPPO proceeds similar to other on-policy algorithms through three distinct phases: data gathering, value target estimation, and value and policy learning. To obtain diverse data, REPPO employs the maximum entropy framework to encourage exploration (Subsubsection 3.1.2). High quality targets come from an adapted version of TD-λ (Subsubsection 3.1.1) to the maximum entropy framework. Finally, to ensure that policies do not collapse and policy learning is stable, REPPO uses KL-constrained policy updates with a schedule that balances entropy-driven exploration (Subsubsection 3.1.4) and policy constraints (Subsubsection 3.1.3).

3.1.1 Stable value learning targets from on-policy data

Off-policy PPG methods like TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018) pair single step Q learning with large replay buffers to stabilize learning. While on-policy algorithms cannot use past policy data, they can instead use multi-step TD targets, which similarly stabilize learning (Fedus et al., 2020). Therefore, multi-step TD-λ targets form the basis for our value learning objectives.

Using TD-λ in on-policy RL has already proven quite useful (Gallici et al., 2024), and it is closely related to Generalized Advantage Estimation (Schulman et al., 2016) which is commonly employed by on-policy methods. GAE combines the TD-λ method with an efficient one-pass algorithm. It is used to reduce the high variance of Monte-Carlo returns by producing smoother estimates of the advantage function $A(x,a) = Q(x,a) - V(x)$. As the gradient of the advantage function and the state-action value function with regard to the action are identical $\nabla_a A(x,a) = \nabla_a Q(x,a) - \cancel{\nabla_a V(x)}$, we directly use TD-$\lambda$ to estimate on-policy state-action value targets.

Finally, using purely on-policy targets allows us to remove several common off-policy stabilization components from the value learning setup. REPPO does not require a pessimism bias, so we can forgo the clipped double Q learning employed by many prior methods (Fujimoto et al., 2018). Tuning pessimistic updates carefully to allow for exploration is a difficult task (Moskovitz et al., 2021), this simplification increases the robustness of our method. We also do not need a target value function copy, since we do not bootstrap our value estimate during each update epoch.

3.1.2 Maximum Entropy Reinforcement Learning

For stable value function learning, diverse data is crucial. To achieve a constant rate of exploration, and prevent the policy from prematurely collapsing to a deterministic function, we leverage the maximum entropy formulation for RL (Ziebart et al., 2008; Levine, 2018). The core aim of the maximum entropy framework is to keep the policy sufficiently stochastic by solving a modified policy objective which penalizes the loss of entropy in the policy distribution. The maximum-entropy policy objective (Levine, 2018) can be defined as

$$J_{\text{ME}}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) + \alpha \mathcal{H}[\pi_\theta(x_t)]\right], \tag{11}$$

where $\mathcal{H}[\pi_\theta(x)]$ is the entropy of the policy evaluated at x, and α is a hyperparameter which trades off reward maximization and entropy maximization. REPPO combines the maximum entropy ob-

jective with TD-λ estimates, resulting in the following target estimate

$$G^{(n)}(x,a) = \sum_{k=t}^{n-1} \gamma^k (r(x_k,a_k) - \alpha \log \pi(a_k|x_k)) + \gamma^n Q(x_n,a_n) \tag{12}$$

$$G^{\lambda}(x,a) = \frac{1}{1-\sum_{t=1}^{N} \lambda^t} \sum_{t=1}^{N} \lambda^t G^{(n)}(x,a), \tag{13}$$

where N is the maximum length of the future trajectory we obtain from the environment for the state-action pair (x,a). Our implementation relies on the efficient backwards pass algorithm presented by Daley & Amato (2019). This is used as a fixed target in each Q learning step in place of the bootstrapped target employed by the SAC algorithm. Our Q learning loss is

$$\mathcal{L}_Q^{\text{REPPO}}(\phi|B) = \frac{1}{|B|} \sum_{i=1}^{|B|} ||Q_\phi(x_i,a_i) - G^{\lambda}(x_i',a')||^2, \tag{14}$$

where $a' \sim \pi(x_i')$ during the environment interaction phase of the environment.

3.1.3 KL-REGULARIZED POLICY LEARNING

The next problem with on-policy policy learning is the size of the policy update. Our value estimate is often only accurate on the data covered by the prior policy. A large policy update can destabilize learning, as the value estimates on the new data are highly erroneous (Kakade & Langford, 2002).

This problem has led to the development of constrained policy update schemes, where the updated policy is prevented from deviating too much from its origin (Peters et al., 2010; Schulman et al., 2015). As the measure of deviation, prior work has almost universally chosen the Kullback-Leibler (KL) divergence, also called the relative entropy, (Peters et al., 2010; Schulman et al., 2015; 2017), as it can be justified theoretically through information geometry (Kakade, 2001; Peters & Schaal, 2008), and is easy to approximate using samples. Formally, given two distributions p and q the KL-divergence can be computed as

$$D_{\text{KL}}[p\|q] = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] \approx \frac{1}{n}\sum_{i=1}^{N} \log \frac{p(x_i)}{q(x_i)},$$

where x_i are iid samples from p. Usually, we either parameterize q or p and keep the other one fixed. If p is the updated distribution, we call this the forward-mode KL, and if we update q, we refer to the backward-mode KL. While most prior methods use the reverse KL to bound the policy update, we find that it leads to a premature collapse of entropy due to its mode-seeking nature. Instead, we use the forward KL, as our unimodal policy makes its mode-averaging behavior benign.

To properly constrain the policy, we also need to develop a robust learning method that can bound the policy deviation and ensure that the maximum entropy objective is achieved. Schulman et al. (2015) and Peters et al. (2010) formulate the KL constrained policy update as a constrained optimization problem. Peters et al. (2010) shows a closed form solution to this problem, while Schulman et al. (2015) uses a conjugate gradient scheme to solve the relaxed optimization problem. Schulman et al. (2017) replaces the Lagrangian formulation with a clipping heuristic. However, clipping can lead to wrong gradient estimates (Ilyas et al., 2020) and in some scenarios the clipping objective fails to bound the policy deviation Wang et al. (2020). Finally, while Schulman et al. (2015) and Schulman et al. (2017) add entropy bonuses to the policy, they do not account for it in the Q function updates.

For REPPO we expand the constrained optimization problem to include both the entropy maximization goal, and the KL constraint

$$\max_{\theta} \quad \mathbb{E}_{\rho_{\pi_{\theta'}}}\left[J_{\text{ME}}(\theta)\right] \tag{15}$$

$$\text{subject to} \quad \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[D_{\text{KL}}\left(\pi_\theta(x) \,\|\, \pi_{\theta'}(x)\right)\right] \leq \text{KL}_{\text{target}} \tag{16}$$

$$\mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[\mathcal{H}[\pi_\theta(x)]\right] \geq \mathcal{H}_{\text{target}}. \tag{17}$$

A similar combination of maximum entropy and KL divergence bound has been explored as an optimization objective in Abdolmaleki et al. (2015) in a black box optimization setting. Here we expand it to the full reinforcement learning problem.

To solve this, we adopt a relaxation which introduces two hyperparameters, α for the entropy coefficient, and β for the KL objective. We constrain the expected entropy and KL over the policies state distribution. Ensuring the constraints for each state individually would introduce too many slack parameters to handle efficiently. Inspired by Haarnoja et al. (2019), REPPO uses a tuning technique to adapt the constraints when the policy update violates them.

3.1.4 Policy Updates and Joint Entropy and KL Distance Tuning

To formulate the constrained objective, we introduce two hyperparameters, $\mathcal{H}_{\text{target}}$ and $\text{KL}_{\text{target}}$, which bound the entropy and KL divergence respectively. The goal of the tuned Lagrangian parameters is to ensure that the policy stays close to these constraints. As we need to ensure that they remain positive, we update them in $\log$ space with a gradient based root finding procedure

$$\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha e^\alpha \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}} \left[(\mathcal{H}[\pi_\theta(x)] - \mathcal{H}_{\text{target}})\right] \tag{18}$$

$$\beta \leftarrow \beta - \eta_\beta \nabla_\beta e^\beta \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}} \left[(\mathcal{H}[\pi_\theta(x)] - \mathcal{H}_{\text{target}})\right] \tag{19}$$

Intuitively, the values are increased if the average policy entropy or KL are below the respective target values, and decreased otherwise. The policy objective for REPPO is

$$\mathcal{L}_\pi^{\text{REPPO}}(\theta|B) = \frac{1}{|B|}\sum_{i=1}^{|B|} -Q(x_i,a_i) + e^\alpha \log\pi(a_i|x_i) + e^\beta \sum_{j=1}^{k} \log \frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)}, \tag{20}$$

where a_j are sampled from the past behavior policy $\pi_{\theta'}$. We also provide code for a clipped variant of the objective

$$\mathcal{L}_\pi^{\text{REPPO}}(\theta|B) = \begin{cases} \frac{1}{|B|}\sum_{i=1}^{|B|} -Q(x_i,a_i) + \alpha\log\pi(a_i|x_i), & \text{if } \sum_{j=1}^{k}\log\frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} < \text{KL}_{\text{target}} \\ e^\beta \sum_{j=1}^{k}\log\frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)}, & \text{otherwise.} \end{cases} \tag{21}$$

This objective enforces the KL bound on a per-state basis instead of in expectation. In practice we have not found a measurable performance difference between both objectives.

Jointly tuning the entropy and KL multipliers is a crucial component of REPPO. Prior algorithms have focused on either tuning the entropy or relative entropy parameters, but as they are deeply linked, it is important to address them together. As the policy entropy and KL are tied, letting the behavior policy collapse to a low entropy could result in a scenario in which the KL constraint prevents any policy updates. Furthermore, the entropy and KL terms are balanced against the scale of the returns in the maximum entropy formulation. As the returns increase over the course of training, keeping the multipliers fixed will cause the model to ignore the constraints over time, accelerating collapse. However, as we tune both in tandem, we find that our setup ensures a steady, constrained amount of slack on the policy to improve while constantly exploring.

3.2 Stable Representation and Value Function Architectures

While the RL algorithm offers a strong foundation to obtain strong surrogate values, we also draw on recent off-policy advances in value function learning that improve training through architecture and loss design. We incorporate three major advancements into REPPO to further stabilize training.

3.2.1 Cross-entropy loss for regression

The first choice is to remove the mean squared error in the critic update with a more robust cross-entropy based loss function. For this, REPPO uses the HL-Gauss loss (Farebrother et al., 2024). This technique was adapted from the distributional C51 algorithm (Bellemare et al., 2017). Surprisingly, algorithms that estimate the return distribution instead of a mean estimate lead to remarkably stable learning algorithms even in deterministic settings. Inspired by this insight and histogram losses for regression (Imani & White, 2018), Farebrother et al. (2024) hypothesize that the benefits are due to the fact that many distributional algorithms use a cross-entropy loss, which is scale invariant. We present the mathematical form of the loss formulation in Appendix A.

REPPO uses HL-Gauss over the categorical C51 loss mostly due to its simpler implementation. In the current literature several works use HL-Gauss and other C51, a final verdict if either one of them is preferable to the other is open. We find that HL-Gauss is a crucial addition, as our ablation experiments show (Subsection 4.5), but other forms of categorical losses could easily work as well.

3.2.2 Layer Normalization

Several recent works (Ball et al., 2023; Yue et al., 2023; Lyle et al., 2024; Nauman et al., 2024a; Hussing et al., 2024; Gallici et al., 2024) have highlighted the importance of layer normalization (Ba et al., 2016) for stable critic learning in reinforcement learning. Gallici et al. (2024) provides a thorough theoretical analysis of the importance of normalization in on-policy learning, while Hussing et al. (2024) focuses on assessing the empirical behavior of networks in off-policy learning with and without normalization. As we operate in an on-policy regime where value function targets are more stable, we find that normalization is not as critical for REPPO as it is for bootstrapped and off-policy methods; yet, we still see performance benefits in most environments from normalization.

3.2.3 Self-prediction Auxiliary tasks

Auxiliary tasks (Jaderberg et al., 2017) are often used to stabilize feature learning in environments with sparse rewards, as the lack of reward signal can prevent the value function from learning meaningful representations (Voelcker et al., 2024a). We find that they are especially impactful when we decrease the number of samples used in each update batch to a minimum (see Subsection 4.5).

A simple yet impactful auxiliary task is latent self prediction (Schwarzer et al., 2021; Voelcker et al., 2024b; Fujimoto et al., 2024). In its simplest form, latent self-prediction is computed by separating the critic into an encoder $\phi : \mathcal{X} \times \mathcal{A} \to \mathcal{Z}$ and a prediction head $f_c : \mathcal{Z} \to \mathbb{R}$. The full critic can then be computed as $Q(x, a) = f_c(\phi(x, a))$. A self-predictive auxiliary loss adds a forward predictive model $f_p : \mathcal{Z} \to \mathcal{Z}$ and trains the encoder and forward model jointly to minimize

$$\mathcal{L}_{\text{aux}}(x_t, a_t, x_{t+1}, a_{t+1}) = \left| f_p(\phi(x_t, a_t)) - \phi(x_{t+1}, a_{t+1}) \right|^2 . \tag{22}$$

As our whole training is on-policy, we do not separate our encoder into a state-dependent and action dependent part as many prior off-policy works have done. Instead we compute the targets on-policy with the behavioral policy and minimize the auxiliary loss jointly with the critic loss.

Overall, the impact of the auxiliary task is the most varied across different environments. In some, it is crucial for learning, while having a detrimental effect in others. We conjecture that the additional learning objective helps retain information in the critic if the reward signal is not informative. In cases where the reward signal is sufficient and the policy gradient direction is easy to estimate, additional training objectives might hurt performance. We encourage practitioners to investigate whether their specific application domain and task benefits from the auxiliary loss.

4 Benchmarking comparison

The primary motivation of this work is to improve the training stability of RL algorithms that learn from on-policy data and thus benefit from the MPS setting. To this end, we evaluate our approach along the following axes: final performance; sample and wall-clock efficiency; stability of the policy improvement process; as well as the memory footprint of the algorithm. In the following subsections, we discuss our experimental results and show strong performance of REPPO along these axes.

4.1 Experimental Setup

We evaluate our algorithm on two major GPU-parallelized benchmark sets: 23 tasks from the mujoco_playground's DMC suite (Zakka et al., 2025), and 8 ManiSkill environments (Tao et al., 2025), which provide us with a locomotion and manipulation benchmarks respectively. We consider a variety of tasks including high-dimensional control, sparse rewards, as well as chaotic dynamics.

As baselines we use the PPO and SAC results provided by the authors of Zakka et al. (2025) and Tao et al. (2025). Given the success of PPO in the MPS setup (Makoviychuk et al., 2021; Kaufmann et al., 2023), we performed an additional hyperparameter search to tune PPO for higher sample

efficiency on the mujoco_playground benchmark. Note that we report all PPO baselines at 50 million as well as the maximum number of time-steps used in the original papers. The latter is always larger than the number of steps we allow for our algorithm. Finally, we use the FastTD3 (Seo et al., 2025), which we benchmark on locomotion DMC tasks. To control for the tradeoff between performance and the memory requirements of the off-policy RL, we train FastTD3 under two memory budgets: with the default size of replay buffer ($10,485,760$ transitions); as the constrained setting in which the FastTD3 buffer uses similar memory size as on-policy algorithms ($32,768$ transitions). We report all implementation details, hyperparameter settings and network architectures in Appendix D.

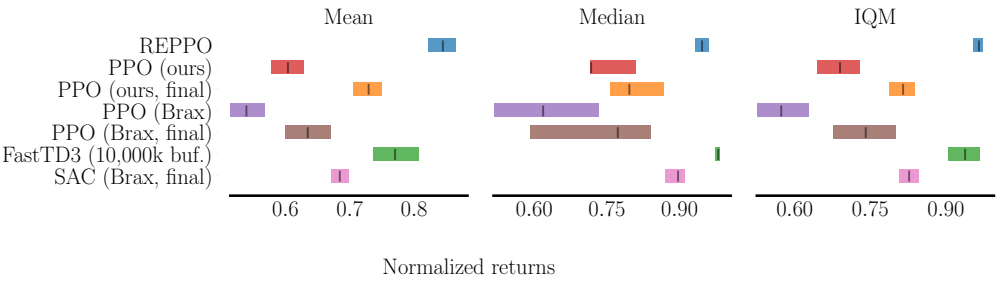
We run each algorithm with multiple random seeds. We use 10 seeds per environment for REPPO, 4 for the baseline PPO methods, and 5 for the FastTD3 results, and report aggregate scores with shaded 95% bootstrapped confidence intervals (Agarwal et al., 2021). Finally, to allow for aggregation across different tasks we normalize the returns of each environments in the mujoco_playground by the maximum return achieved by any algorithm. For ManiSkill, we report success rates, which have a natural interpretation and can be compared between different tasks.

4.2 Final Performance and Sample Efficiency

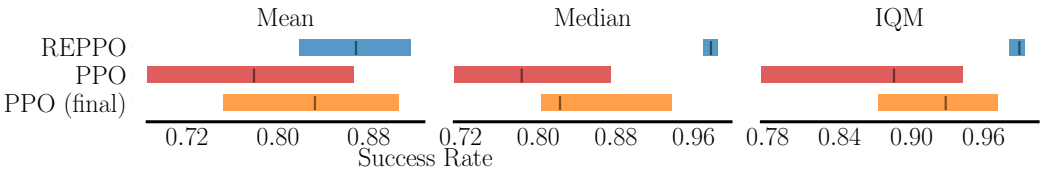
We first investigate the performance of policies trained using REPPO. We report aggregate performance evaluations at the end of training on both benchmarks in Figure 2. For both benchmarks, we also provide the corresponding training curves in Figure 3. Finally, we chose four representative environments from each benchmark suite to present in Figure 4. These are chosen from varying levels of difficulty and include the one environment in which a baseline beats REPPO in both suites.

The aggregate results shown in Figure 2 and Figure 3 indicate that our proposed method achieves statistically significant performance improvements over PPO and SAC, as well as similar performance to FastTD3 despite REPPO being fully on-policy. Although these results are most pronounced in locomotion tasks, ManiSkill manipulation results show significant performance benefits over PPO in terms of outlier-robust metrics (Agarwal et al., 2021).

The per-environment results (Figure 4) highlight the stability and sample efficiency of REPPO specifically and pathwise policy gradients in general. Our proposed approach improves its pol-



(a) Aggregate performance metrics on the mujoco_playground DeepMind Control Suite benchmark. We compare both REPPO and the tuned PPO baseline at 50 million environment steps. We also report the performance of the Brax PPO and SAC implementations provided by Zakka et al. (2025). Zakka et al. (2025).



(b) Aggregate success on the ManiSkill3 benchmark (Tao et al., 2025). We compare REPPO against a PPO baseline provided by Tao et al. (2025) at 50 million environment steps. As some environments take more then 50 million steps for PPO to achieve strong performance, we report the final performance at 100 million steps. While the mean confidence intervals are very broad, REPPO performs strongly on the IQM and median metrics.

Figure 2: Aggregate performance comparison on (a) mujoco_playground DMC and (b) ManiSkill3.

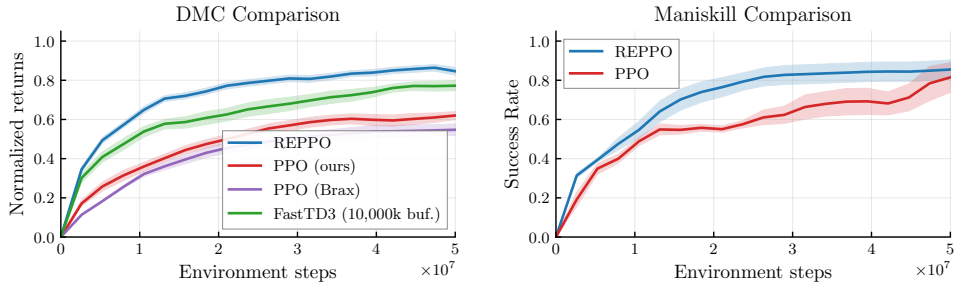
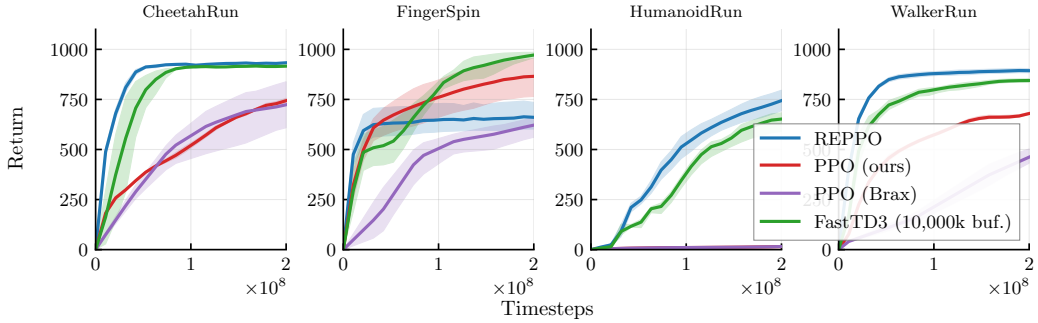


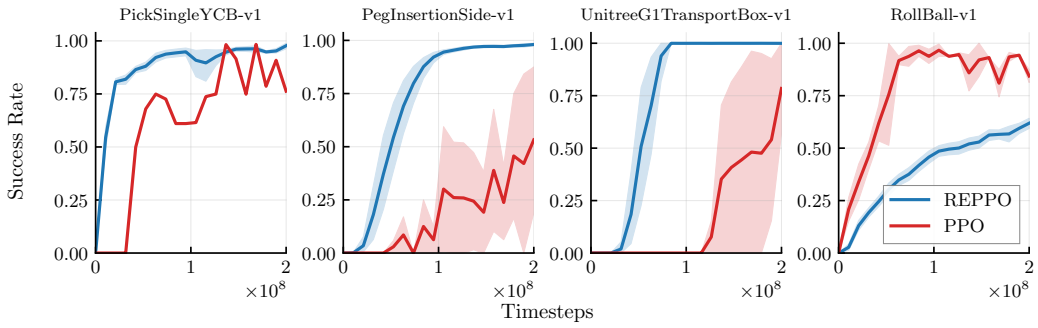
Figure 3: Aggregate sample efficiency curves for the benchmark environments. Settings are identical to those in Figure 2. REPPO achieves higher performance at a faster rate in both benchmarks.

icy at a significantly faster rate than the score-based PPO. However, we do find tasks in which PPO outperforms our proposed method. This suggests that there are important axes on which REPPO can still be improved, such as exploration and premature convergence.

Finally, we find that PPO struggles with high-dimensional tasks like HumanoidRun, despite using large batch sizes for policy gradient variance reduction. Similarly, we find that despite the approximate trust region policy update performed in PPO, applying the policy gradients often leads to a decrease in performance and erratic training. This erratic behavior is highly reminiscent of the behavior we observed earlier in Figure 1 for score-based policy gradients. In contrast, REPPO improves in a more stable way, generally leading to lower variance between individual seeds.



(a) Evaluation on individual DMC environments. REPPO consistently starts learning early and achieves high performance quickly even on difficult tasks such as HumanoidRun. In rare cases such as FingerSpin, REPPO converges to a suboptimal solution.



(b) Evaluation on individual ManiSkill environments. REPPO shows consistent and fast learning while PPO's success rates are very erratic and have high variance.

Figure 4: Evaluation on individual environments for (a) mujoco_playground DMC and (b) ManiSkill

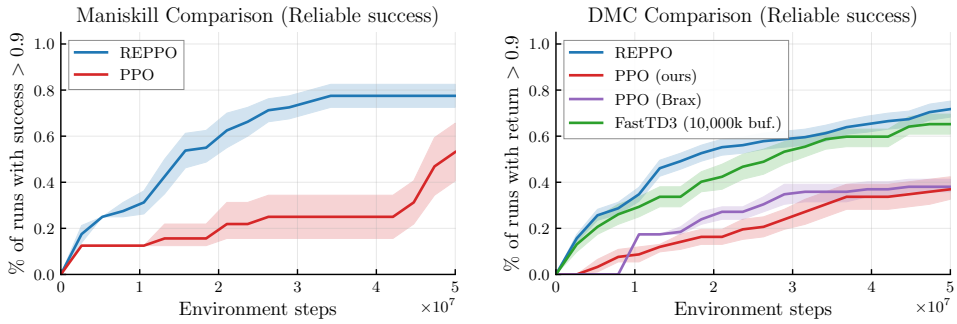


Figure 5: Fraction of runs that achieve reliable performance as measured by our metric for policy stability and reliability. REPPO's immediately starts achieving high performance in some runs and the number gradually increases indicating stable learning. PPO struggles to achieve high performance initially and to maintain high performance throughout training.

4.3 Stable Policy Improvement

We further investigate the stability of policy improvements performed via score-based and pathwise policy gradients. Here, our rationale is that performing policy improvements should not lead to a large decreases in performance. We formalize this notion into a metric that we refer to as "reliable performance". We say that the execution of an algorithm is reliably performant once performance levels reach a fixed threshold τ and afterward *never fall below it*. At each timestep, we then count how many runs currently meet these criteria. This metric is inspired by practical considerations in which a deployed algorithm should not suddenly start to fail simply because we continue training it. We report the percentage of reliably successful runs for both REPPO and PPO in Figure 5.

On both DMC and ManiSkill benchmarks, we observe that REPPO policy improvement quickly leads to reliable performance improvements, with success rates and returns gradually increasing. At the end of the training, roughly four out of five runs have reliably reached the performance threshold of $\tau = 0.9$ and have not fallen below again. In comparison, the ultimate number of reliably performant runs on PPO is approximately 40 percentage points smaller than on REPPO. Furthermore, we observe pronounced differences in the sample efficiency of two policy gradient strategies, with PPO taking five to ten million environment interactions before any runs achieve reliable performance. These results highlight that despite using a biased surrogate value model trained in on-policy setup, REPPO pathwise policy gradient approach leads to stable policy improvement over long training.

4.4 Wall-clock Time

In this subsection, we investigate the wall-clock performance of our proposed approach. Wall-clock is a particularly useful metric when considering isolated training in simulation, as it highlights the practical utility of an algorithm. Fast algorithms allow for efficient hyperparameter search and fast training. Unfortunately, rigorous wall-clock time measurement is a difficult, as many factors impact the wall-clock time of an algorithm and it is hard to reproduce on different machines. We further discuss the difficulties and approaches used when measuring wall clock time across different implementation frameworks in Appendix B.

In Figure 6, we investigate the wall-clock performance of our approach when compared to PPO, with Jax jit functionality used to compile

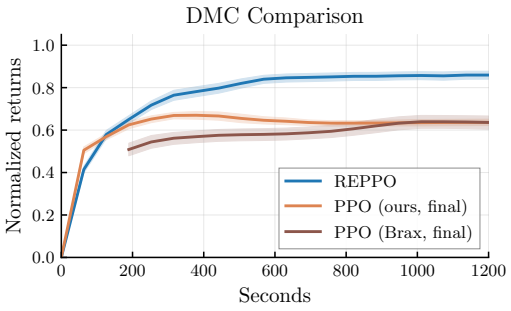


Figure 6: Wall-clock time comparison of REPPO against common PPO implementations. REPPO is similarly fast as PPO but gets higher return. Each experiment was run on a dedicated L40s compute node.

the both the algorithms and environment end-to-end. REPPO's jax-based implementation is on par with the speed of standard PPO implementations (Lu et al., 2022; Freeman et al., 2021). Note that the computational cost per update step is higher for REPPO than for PPO, as we use larger default networks, and propagate the gradient through the critic-actor chain. However, our proposed approach converges to a better performing policy at roughly similar speeds. More precisely, both algorithms converge after roughly 600-800 seconds but REPPO obtains roughly 33% improved normalized returns. As such, we observe that the improved sample efficiency of pathwise gradients can effectively offset the increase in computation stemming from policy updates via surrogate value model, leading to improvements in wall-clock efficiency over score-based PPO.

4.5 Design Ablations

We run ablation experiments investigating the impact of the design components used in REPPO. In these experiments, we remove the cross-entropy loss via HL-Gauss, layer normalization, the auxiliary self-predictive loss, or the KL regularization of the policy updates. To understand the importance of each component for on-policy learning we conduct these ablations for two scales of batch sizes - the default $131,072$ on-policy transitions, as well as the smaller batch size of $32,768$.

As shown in Figure 7, our results indicate that both the KL regularization of the policy updates and the categorical Q-learning via HL-Gauss are necessary to achieve strong performance independent of the size of the on-policy data used to update our model. We find that the KL divergence is the only component that, when removed, leads to a decrease in performance below the levels of PPO, which clarifies the central importance of relative entropy regularization for REPPO. Removing normalization has minor negative effects on performance which become worse at smaller buffer sizes. This is consistent with the literature on layer normalization in RL. Similarly, the auxiliary self-predictive loss has significantly more impact on performance when the batch size becomes smaller. We note that auxiliary loss has an inconsistent impact on the training generally, where it is strongly beneficial in some environments, but harmful in others.

4.6 Memory demands

Our final result concerns itself with memory demands. Recent advances in off-policy algorithms have shown great performance when large buffer sizes are available (Seo et al., 2025). When dealing with complex observations such as images, on-policy algorithms which do not require storing past data have a large advantage. In terms of data storage requirements, our algorithm is comparable with PPO, yet it remains to answer how well REPPO compares to algorithms that are allowed to store a large amount of data. For this, we compare against the recent, very strongly performing FastTD3 (Seo et al., 2025) which also uses GPU-parallelized environments but operates off-policy.

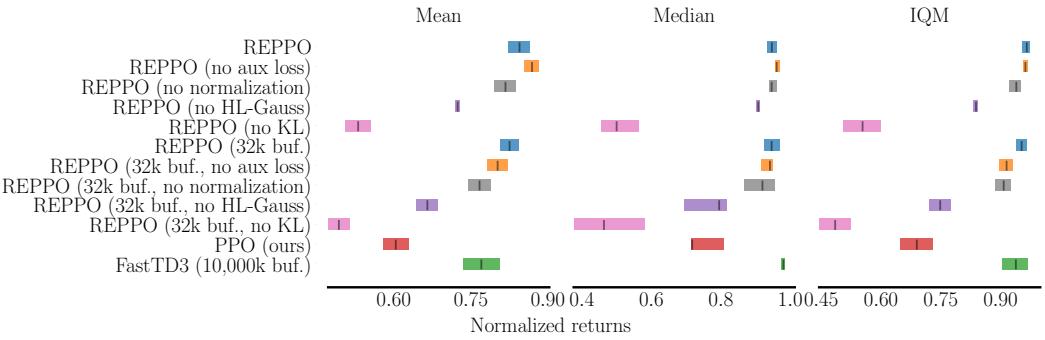


Figure 7: Ablation on components and data size on the DMC benchmark. The larger size corresponds to 128×1024 samples, while the smaller corresponds to 32×1024 samples. Both values are significantly smaller than the replay buffer sizes used in standard off-policy RL algorithms like SAC and FastTD3. The HL-Gauss loss and KL regularization provide a clear benefit at both data scales. The normalization and auxiliary loss become more important when less data is available, highlighting that some stability problems can also be overcome with scaling data.

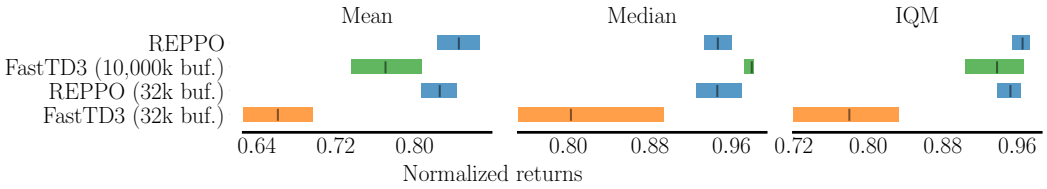


Figure 8: Comparison of aggregate performance between REPPO and FastTD3. REPPO is competitive with the large buffer FastTD3 version and outperforms FastTD3 when memory is limited.

We compare REPPO against the original FastTD3 and we also re-run FastTD3 with access to a significantly smaller buffer equivalent to the REPPO buffer. We report the results in Figure 8.

The results demonstrate that REPPO is on par or better in terms of performance on mean and IQM with the FastTD3 approach. This is despite the fact that REPPO uses a buffer that is two to three orders of magnitude smaller. When decreasing the buffer size of FastTD3, the algorithm's performance drops by a large margin while REPPO is barely affected by a smaller buffer. We find that FastTD3 with a smaller buffer can retain performance on lower dimensional, easier tasks but suffers on harder tasks that may be of greater interest in practice. In summary, REPPO is competetive with recent advances in off-policy learning with significantly lower memory and storage requirements.

5 Related Work

Stabilizing On-Policy RL A fundamental issue with score-based approaches is their instability. Therefore, various improvements to decrease gradient variance have been considered. Some works have noted the difficulty of representation learning and have addressed this via decoupling the training of value and policy (Cobbe et al., 2021; Aitchison & Sweetser, 2022). Moalla et al. (2024) note that feature learning problems can result from representation collapse, which can be mitigated using auxiliary losses. There are also efforts to reduce the variance of gradients, e.g. by finding a policy that minimizes the variance of the importance sampling factor (Papini et al., 2024) or modifying the loss to ensure tighter total variational distance constraints (Xie et al., 2025). Akrour et al. (2019) propose to project the policy onto the trust-region to sidestep the difficulty associated with clipping. However, none of these address the issue at the root by introducing a first-order return gradient.

Otto et al. (2021) propose to replace the various trust-region enforcement methods such as line-search or clipping with differentiable trust-region layers in the policy neural network architecture. While our method is slightly more general, as we make no assumption on the form of the policy (aside from assuming gradient propagation through the sampling process is possible), trust-region layers could easily be combined with REPPO for appropriate policy parameterizations.

Incorporating ground-truth gradient signal to stabilize training has been studied, both for dynamical systems (Son et al., 2023) and differentiable robotics simulation (Mora et al., 2021; Xu et al., 2022; Georgiev et al., 2024). However, access to a ground-truth gradient requires custom simulators, and in contact-rich tasks, surrogate models can provide smoother gradients (Suh et al., 2022b).

Work on GPU-parallelized On-policy RL With the parallelization of many benchmarks on GPUs (Makoviychuk et al., 2021; Zakka et al., 2025; Tao et al., 2025), massively-parallel on-policy RL has become quite popular. While these environments provide simulation testbeds, algorithms trained in such environments have shown to transfer to real-robots, allowing us to train them in minutes rather than days (Rudin et al., 2022).

Hybridizing Off-policy and On-policy RL methods Most closely to our work, Parallel Q Networks (PQN) (Gallici et al., 2024) was established by using standard discrete action-space off-policy techniques in the MPS setting. While our work shares several important features with this method, we find that our additional insights on KL regularization and tuning is crucial for adapting the concept to continuous action spaces. In the future, unifying both algorithms in a similar vein to (Fujimoto et al., 2024) is an important step to simplify the landscape of algorithms.

Other methods, such as Parallel Q-Learning (Li et al., 2023) and FastTD3 (Seo et al., 2025) also attempt to use deterministic policy gradient algorithms in the MPS setting, but still remain off-policy. This has two major drawbacks compared to our work. The methods require very large replay buffers, which can either limit the speed if data needs to be stored in regular CPU memory, or require very large and expensive GPUs. In addition, the off-policy nature of these methods requires stabilizing techniques such as clipped double Q learning, which has been shown to prevent exploration.

KL-based RL Finally, other works also build on top of the relative entropy policy search (Peters et al., 2010). Maximum A Posteriori Policy Optimization (MPO) (Abdolmaleki et al., 2018) and Variational MPO (Song et al., 2019) both leverage SAC style maximum entropy objectives and use KL constraints to prevent policy divergence. However, both methods use off-policy data together with importance sampling, which we forgo, do not tune the KL and entropy parameters, and crucially do not make use of the deterministic policy gradient.

Going beyond relative entropy, the KL-based constraint formulation has been generalized to include the class of mirror descent algorithms (Grudzien et al., 2022; Tomar et al., 2022). In addition, Lu et al. (2022) meta-learns a constraint to automatically discover novel RL algorithms. These advancements are largely orthogonal to our work and can be incorporated into REPPO in the future.

Instability in Off-policy RL Our method furthermore adapts many design decisions from recent off-policy literature. Among these are layer normalizations, which have been studied by Nauman et al. (2024a); Hussing et al. (2024); Nauman et al. (2024b); Gallici et al. (2024), auxiliary tasks (Jaderberg et al., 2017; Schwarzer et al., 2021; 2023; Tang et al., 2023; Voelcker et al., 2024b; Ni et al., 2024), and HL-Gauss (Farebrother et al., 2024), variants of which have been used by Hafner et al. (2021); Hansen et al. (2024); Voelcker et al. (2025). Beyond these, there are several other works which investigate architectures for stable off-policy value learning, such as Nauman et al. (2024b); Lee et al. (2025a;b). A similar method to our KL regularization tuning objective has been used by (Nauman & Cygan, 2025) to build an exploratory optimistic actor. While the technique is very similar, we employ it in the context of the trust-region update, and show the importance of jointly tuning the entropy and KL parameters. Finally, there are several papers which investigate the impact of continual learning in off-policy reinforcement learning, including issues such as out-of-distribution misgeneralization (Voelcker et al., 2025), plasticity loss (Nikishin et al., 2022; D'Oro et al., 2023; Lyle et al., 2023; Abbas et al., 2023). Since many of these works focus specifically on improving issues inherent in the off-policy setting, we did not evaluate all of these changes in REPPO. However, rigorously evaluating what network architectures and stabilization methods can help to further improve the online regime is an exciting avenue for future work.

6 Conclusion and avenues for future work

In this paper we present REPPO, a highly performant but efficient on-policy algorithm that leverages pathwise instead of score-based policy gradients. By balancing entropic exploration and KL-constraints, and incorporating recent advances in neural network value function learning, REPPO is able to learn a high-quality surrogate function sufficient for reliable gradient estimation. As a result, the algorithm outperforms PPO on two GPU-parallelized benchmarks in terms of final return, sample efficiency and reliability while being on par in terms of wall-clock time. In addition, the algorithm does not require storing large amount of data making it competitive with recent advances in off-policy RL while requiring orders of magnitude lower amounts of memory.

We believe that REPPO provides a basis that will allow other researchers to conduct impactful research without suffering from the instability that comes from score-based approaches or the common challenges in off-policy learning. Ultimately, this should lead to more reliable generation of scientific insights but also more trustworthy deployment of RL algorithms in the wild.

As our method opens a new area for algorithmic development, it leaves open many exciting avenues for future work. As Seo et al. (2025) shows, using replay buffers can be beneficial to stabilize learning as well. This opens the question if our Q learning objective can be expanded to use both on- and off-policy data to maximize performance while minimizing memory requirements. Furthermore, the wide literature on improvements on PPO, such as learned constraint objectives (Lu et al., 2022) could be incorporated into REPPO. Finally, better architectures such as Nauman et al. (2024b), Lee

et al. (2025a), Otto et al. (2021) might be transferable to our algorithm and the rich literature on architectural improvements in off-policy RL can be expanded to include on-policy value learning.

ACKNOWLEDGMENTS

MH and EE were partially supported by the Army Research Office under MURI award W911NF201-0080 and the DARPA Triage Challenge under award HR00112420305. AMF acknowledges the support of NSERC through the Discovery Grant program [2021-03701]. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. Pieter Abbeel holds concurrent appointments as a Professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, the US Army, the US government, or the Canadian government.

The authors thank Stone Tao for providing results for and help with the maniskill baseline, and Kevin Zakka for results for and support with mujoco_playground. We thank Younggyo Seo and co-authors for the FastTD3 codebase, which provides the scaffolding for REPPO's torch implementation, and Chris Lu and co-authors for the purejaxrl library which forms the basis of our jax implementation.

REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. In *Proceedings of the Conference on Lifelong Learning Agents*, 2023.

Abbas Abdolmaleki, Rudolf Lioutikov, Jan R Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. *Advances in Neural Information Processing Systems*, 2015.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, 2021.

Matthew Aitchison and Penny Sweetser. DNA: Proximal policy optimization with a dual network architecture. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Riad Akrour, Joni Pajarinen, Jan Peters, and Gerhard Neumann. Projections for approximate policy iteration algorithms. In *International Conference on Machine Learning*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *ArXiv*, volume abs/1607.06450, 2016.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning*. Springer, 1995.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the International Conference on Machine Learning*, 2023.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Brett Daley and Christopher Amato. Reconciling λ-returns with experience replay. In *Advances in Neural Information Processing Systems*, 2019.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Proceedings of the International Conference on Learning Representations*, 2023.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *Proceedings of the International Conference on Machine Learning*, 2024.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *Proceedings of the International Conference on Machine Learning*, 2020.

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.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning*, 2018.

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. In *International Conference on Learning Representations*, 2024.

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*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In *International Conference on Machine Learning*. PMLR, 2024.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5, 2004.

Jakub Grudzien, Christian A Schroeder De Witt, and Jakob Foerster. Mirror learning: A unifying framework of policy optimisation. In *International Conference on Machine Learning*, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning*, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *Proceedings of the International Conference on Learning Representations*, 2021.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth Proceedings of the International Conference on Learning Representations*, 2024.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

Marcel Hussing, Claas Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton. Dissecting deep RL with high update ratios: Combatting value divergence. In *Reinforcement Learning Conference*, 2024.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the International Conference on Learning Representations*, 2017.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 2001.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2021.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R. Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025b.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Jiajin Li, Baoxiang Wang, and Shengyu Zhang. Policy optimization with second-order advantage information. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5038–5044, 2018.

Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel q-learning: Scaling off-policy reinforcement learning under massively parallel simulation. In *Proceedings of the International Conference on Machine Learning*, pp. 19440–19459, 2023.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2016.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *Proceedings of the International Conference on Machine Learning*, 2023.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado Van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks, 2024. Publication Title: ArXiv Volume: abs/2402.18762.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. No representation, no trust: Connecting representation, collapse, and trust issues in PPO. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

Miguel Angel Zamora Mora, Momchil Peychev, Sehoon Ha, Martin Vechev, and Stelian Coros. Pods: Policy optimization via differentiable simulation. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Michal Nauman and Marek Cygan. Decoupled policy actor-critic: Bridging pessimism and risk awareness in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzcinski, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2024a.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. In *Advances in Neural Information Processing Systems*, 2024b.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive RL. In *Proceedings of the International Conference on Learning Representations*, 2024.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2022.

Fabian Otto, Philipp Becker, Vien Anh Ngo, Hanna Carolin Maria Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qYZD-AO1Vn.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Matteo Papini, Giorgio Manganini, Alberto Maria Metelli, and Marcello Restelli. Policy gradient with active importance sampling. *Reinforcement Learning Journal*, 2:645–675, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019.

Jan Peters and Stefan Schaal. Natural actor-critic. In *Neurocomputing*, volume 71. Elsevier, 2008.

Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

Nate Rahn, Pierluca D'Oro, Harley Wiltzer, Pierre-Luc Bacon, and Marc Bellemare. Policy optimization in a noisy neighborhood: On return landscapes in continuous control. *Advances in Neural Information Processing Systems*, 36:30618–30640, 2023.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pp. 91–100. PMLR, 2022.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *International Conference for Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *Proceedings of the International Conference on Learning Representations*, 2021.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *Proceedings of the International Conference on Machine Learning*, 2023.

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. FastTD3: Simple, fast, and capable reinforcement learning for humanoid control, 2025.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning*, 2014.

Sanghyun Son, Laura Yu Zheng, Ryan Sullivan, Yi-Ling Qiao, and Ming Lin. Gradient informed proximal policy optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=oFaLc6fHSt.

H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin Riedmiller, and Matthew M. Botvinick. V-MPO: On-Policy Maximum a Posteriori Policy Optimization for Discrete and Continuous Control. In *International conference on learning representations*, 2019.

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, 2022a.

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *Proceedings of the International Conference on Machine Learning*, 2022b.

Richard S Sutton. Learning to predict by the methods of temporal differences. In *Machine learning*, volume 3. Springer, 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2nd edition, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. In *Journal of Machine Learning Research*, volume 17. MIT Press, 2016.

Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Ávila Pires, Yash Chandak, Rémi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, and others. Understanding self-predictive learning for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2023.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Connectionist Models Summer School*, 1993.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. In *International Conference on Learning Representations*, 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Hado Van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, 2010.

Claas Voelcker, Marcel Hussing, and Eric Eaton. Can we hop in general? a discussion of benchmark selection and design using the hopper environment. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024a.

Claas Voelcker, Tyler Kastner, Igor Gilitschenski, and Amir-massoud Farahmand. When does self-prediction help? understanding auxiliary tasks in reinforcement learning. In *Reinforcement Learning Conference*, 2024b.

Claas Voelcker, Marcel Hussing, Eric Eaton, Amir-massoud Farahmand, and Igor Gilitschenski. MAD-TD: Model-augmented data stabilizes high update ratio RL. In *Proceedings of the International Conference on Learning Representations*, 2025.

Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence*, 2020.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Forty-second International Conference on Machine Learning*, 2025.

Jie Xu, Miles Macklin, Viktor Makoviychuk, Yashraj Narang, Animesh Garg, Fabio Ramos, and Wojciech Matusik. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZSKRQMvttc.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving q-value divergence in offline-RL. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo playground: An open-source framework for GPU-accelerated robot learning and sim-to-real transfer., 2025. URL https://github.com/google-deepmind/mujoco_playground.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, 2008.

A HL-GAUSS EQUATIONS

Given a regression target y and a function approximation $f(x)$, HL-Gauss transforms the regression problem into a cross-entropy minimization. The regression target is reparameterized into a histogram approximation hist of $\mathcal{N}(y, \sigma)$, with a fixed σ chosen heuristically. The number of histogram bins h and minimum and maximum values are hyperparameters. Let $\mathrm{hist}(y)_i$ be the probability value of the histogram at the i-th bucket. The function approximation has an h-dimensional output vector of logits. Then the loss function is

$$\mathrm{HL}(y, f(x)) = \sum_{i=1}^{h} \mathrm{hist}(y)_i \cdot \log \frac{\exp f(x)_i}{\sum_{j=1}^{h} \exp f(x)_j} \quad .$$

The continuous prediction can be recovered by evaluating

$$\hat{y} = \mathbb{E}[\mathrm{hist}(f(x))] = \langle \mathrm{hist}(f(x)), \mathrm{vec}(\min, \max, h) \rangle,$$

where $\mathrm{vec}(\min, \max, h)$ is a vector with the center values of each bin ranging from $\min$ to $\max$.

B WALLCLOCK MEASUREMENT CONSIDERATIONS

Measuring wall-clock time has become a popular way of highlighting the practical utility of an algorithm as it allows us to quickly deploy new models and iterate on ideas. Rigorous wall-clock time measurement is a difficult topic, as many factors impact the wall-clock time of an algorithm.

We chose to not compare the jax and torch-based versions head to head as we found significant runtime differences on different hardware, and the different compilation philosophies lead to different benefits and drawbacks. For example, jax' full jit-compilation trades a much larger initial overhead for significantly faster execution, which can amortize itself depending on the number of timesteps taken. Especially the tanh-squashed log probability computation cannot be offloaded into an efficient kernel without providing one manually, which we have not done. Doing so would be an excellent way to speed up future SAC derived algorithms. This is not a concern for jax, due to the fact that all kernels are statically compiled when the program is first executed and so the CPU is under much lower load.

Instead of raw wall-clock time measurements, which can vary massively across framework and hardware, we recommend that the community treat the question of wall-clock time more carefully. While the actual time for an experiment can be of massive importance from a practical point of view, the advantages and limitations of current frameworks can obscure exciting directions for future work. For example REPPO is highly competitive with PPO when implemented in jax, but struggles somewhat in torch due to framework specific design choices.

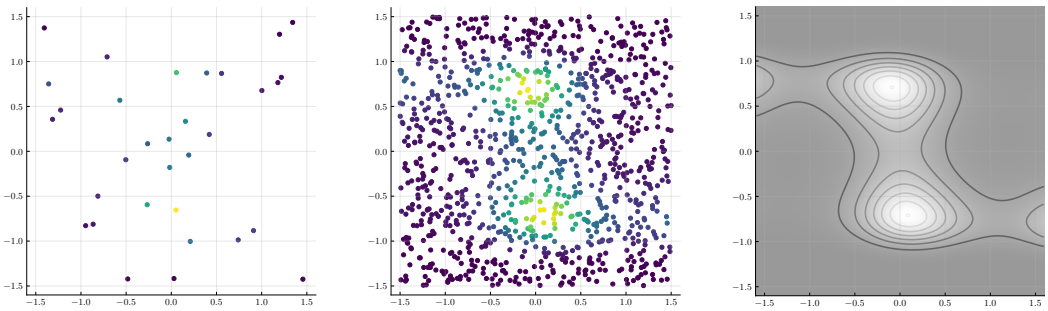


Figure 9: Samples used to train the surrogate function. On the left, we visualize the 32 sample dataset to train the weak surrogate function, in the middle the 1024 datapoints to train the strong, and on the right the full objective function.

C Implementation details and hyperparameters

C.1 Toy example

To obtain the gradient descent comparison in Subsection 2.3 we used the 6-hump camel function, a standard benchmark in optimization. As our goal was not to show the difficulties of learning with multiple optima, which affect any gradient-based optimization procedure, but rather smoothness of convergence, we initialized all runs close to the global minimum. The surrogate functions were small three layer, 16 unit MLPs. To obtain a strong and a weak version, we used differing numbers of samples, visualized in Figure 9. Every algorithm was trained with five samples from the policy at every iteration. Finally, we tested several learning rates. We chose a learning rate which allows the ground-truth pathwise gradient to learn reliably. If a smaller gradient step size is chose, the Monte-Carlo estimator converges more reliably, at the cost of significant additional computation. We also tested subtracting a running average mean as a control variate from the Monte-Carlo estimate. While this reduced variance significantly, it was still very easy to destabilize the algorithm by choosing a larger step size or less data samples.

In total, our experiments further highlight a well known fact in gradient-based optimization: while a MC-based gradient algorithm can be tuned for strong performance, it is often extremely dependent on finding a very good set of hyperparameters. In contrast, pathwise estimators seem to work much more reliably across a wider range of hyperparameters, which corroborates our insights on REPPO hyperparameters robustly transfering across environemnts and benchmark suites.

D REPPO Main Experiments

In addition to the details laid out in the main paper, we briefly introduce the architecture and additional design decisions, as well as default hyperparameter settings.

The architecture for both critic encoder and heads, as well as the actor, consists of several normalized linear layer blocks. As the activation function, we use silu/swift. As the optimizer, we use Adam. We experimented with weight decay and learning rate schedules, but found them to be harmful to performance. Hyperparameters are summarized in Table 1. We tune the discount factor γ and the minimum and maximum values for the HL-Gauss representation automatically for each environment, similar to previous work (Hansen et al., 2024).

For all environments, we use observation normalization statistics computed as a simple running average of mean and standard deviation. We found this to be important for performance, similar as in other on policy algorithms. Since we do not hold data in a replay buffer, we do not need to account for environment normalization in a specialized manner, and can simply use an environment wrapper.

For more exact details on the architecture we refer to interested readers to the codebase.

Finally, we provide sample efficiency curves per environment in Figure 10, Figure 11, and Figure 12.

Environment	
total time steps	$50,000,000$
n envs	1024
n steps	128
$\mathrm{KL}_{\mathrm{target}}$	0.1
Optimization	
n epochs	8
n mini batches	64
batch size	$\frac{\text{n envs} \times \text{n steps}}{\text{n mini batches}} = 2048$
lr	$3e-4$
maximum grad norm	0.5
Problem Discount	
γ	$1 - \frac{10}{\max \text{env steps}}$
λ	0.95

Critic Architecture	
critic hidden dim	512
vmin	$\frac{1}{1-\gamma} \min r$
vmax	$\frac{1}{1-\gamma} \max r$
num HL-Gauss bins	151
num critic encoder layers	2
num critic head layers	2
num critic pred layers	2
Actor Architecture	
actor hidden dim	512
num actor layers	3
RL Loss	
β start	0.01
$\mathrm{KL}_{\mathrm{target}}$	0.1
α start	0.01
$\mathcal{H}_{\mathrm{target}}$	$0.5 \times \dim \mathcal{A}$
aux loss mult	1.0

Table 1: Default REPPO hyperparameters

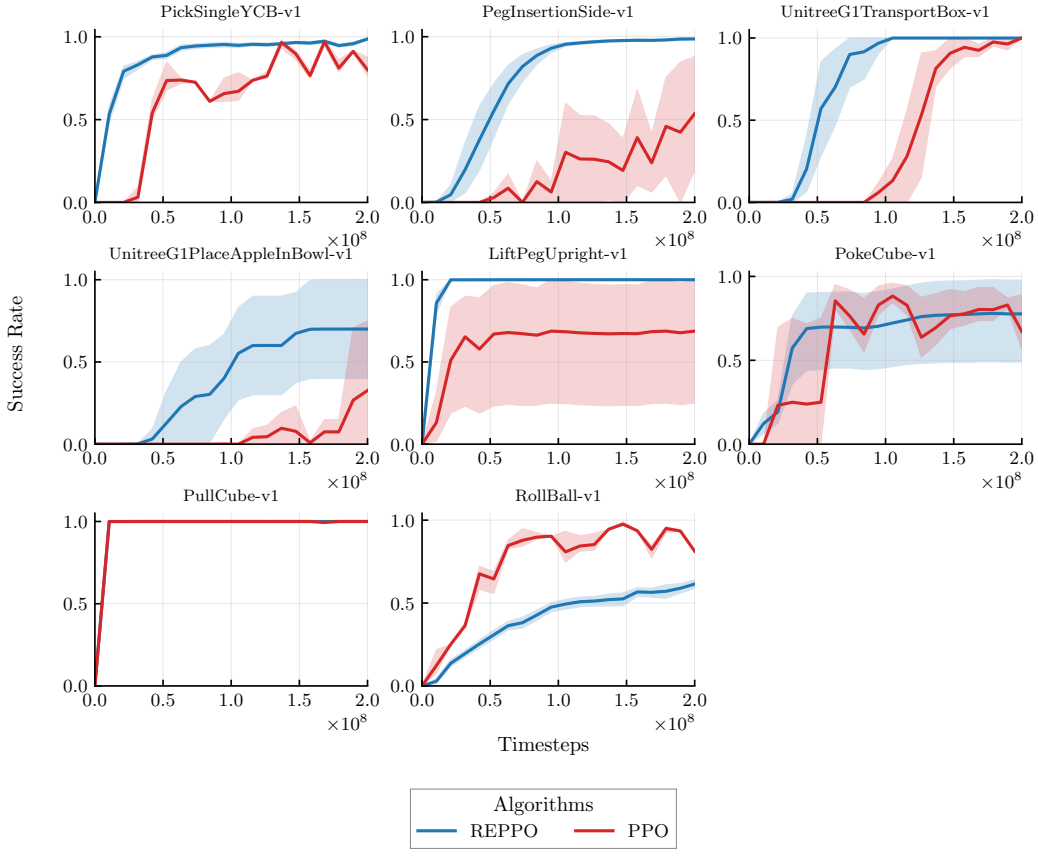


Figure 10: Per-environment results on the ManiSkill suite

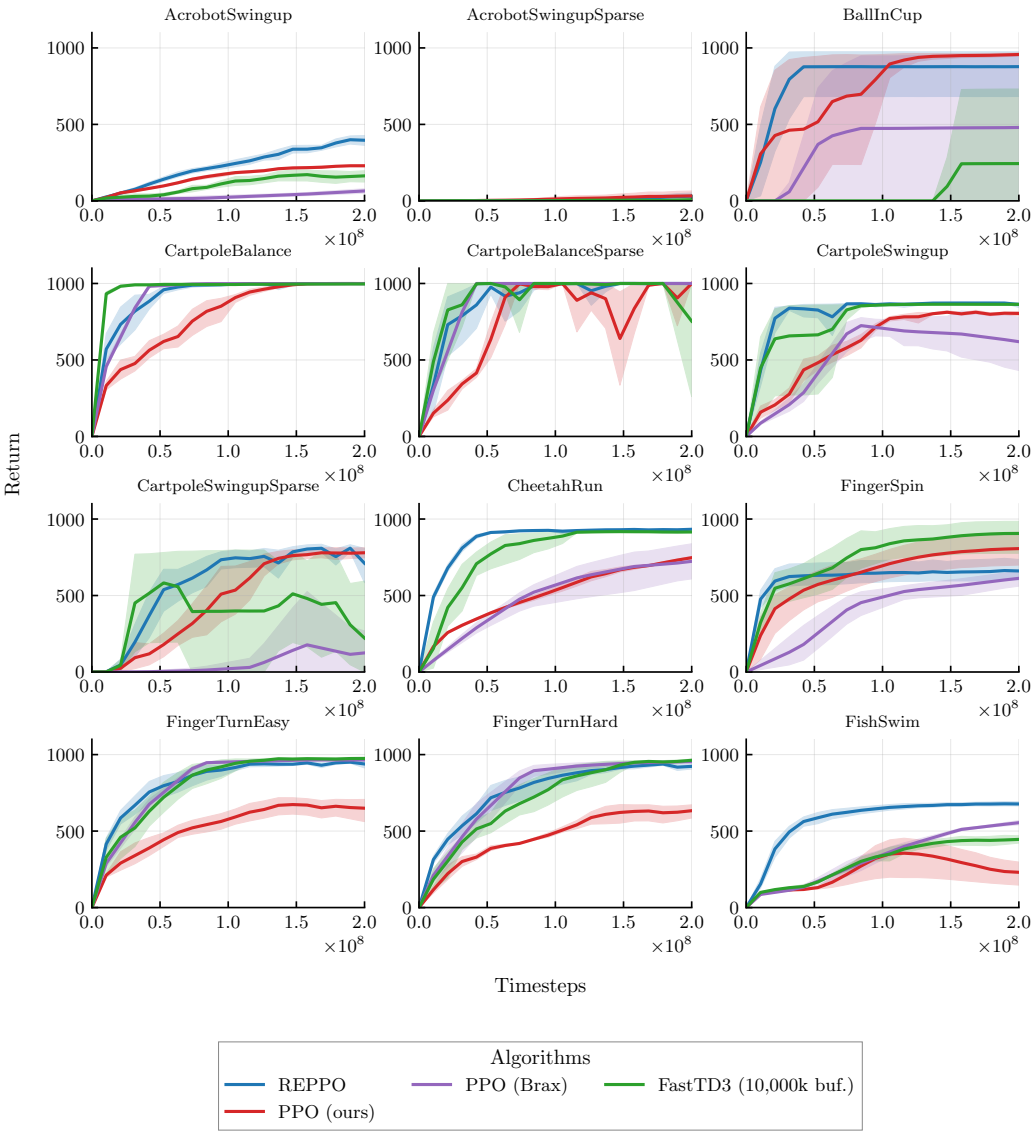


Figure 11: Per-environment results on the mujoco_playground DMC suite

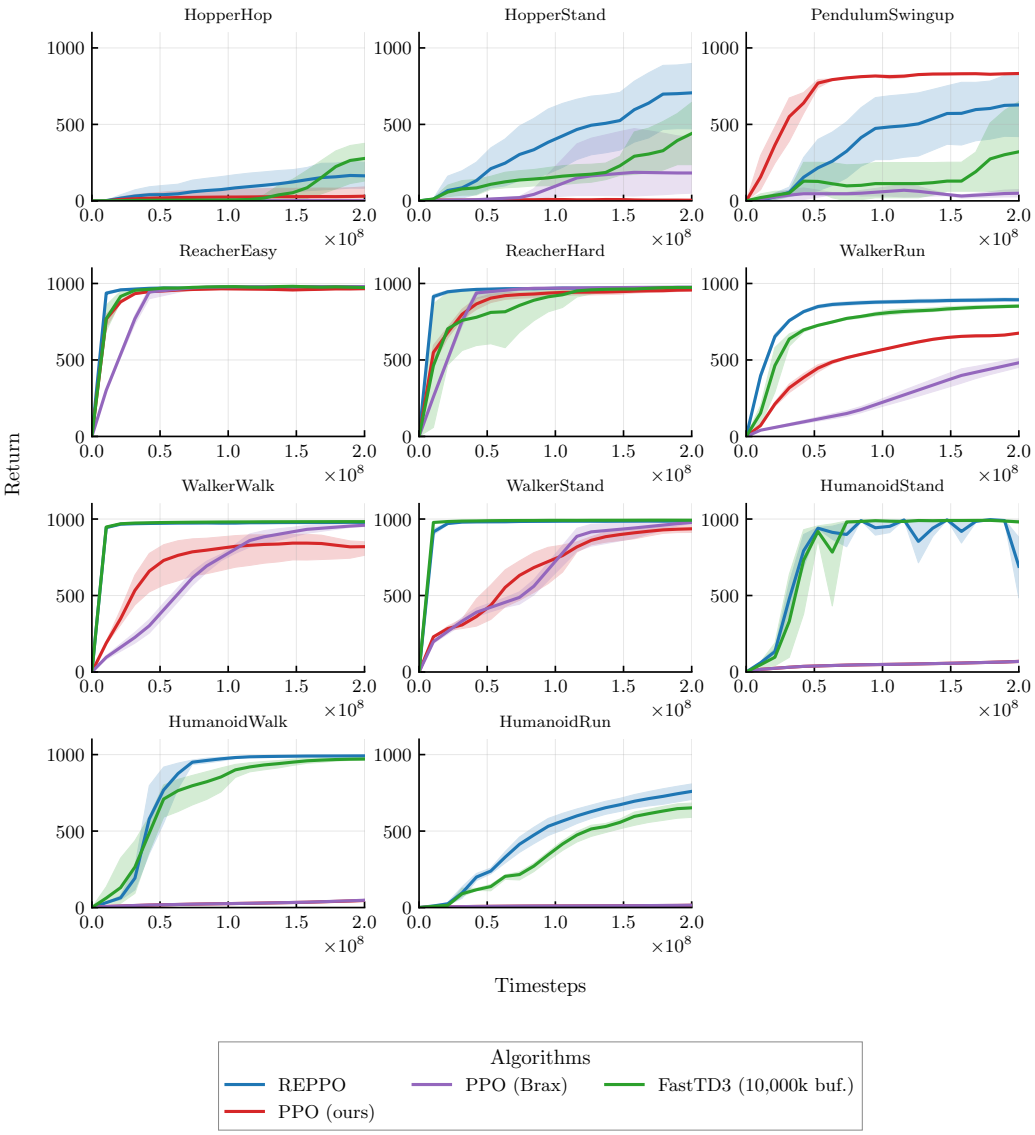


Figure 12: Per-environment results on the mujoco_playground DMC suite