RELATIVE ENTROPY PATHWISE POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Score-function based methods for policy learning, such as REINFORCE and PPO, have delivered strong results in game-playing and robotics, yet their high variance often undermines training stability. ~~Using pathwise policy gradients, i.e. computing a derivative by differentiating the objective function~~Improving a policy through state-action value functions, e.g. by differentiating Q with regard to the policy, alleviates the variance issues. However, ~~they require~~ this requires an accurate action-conditioned value function, which is notoriously hard to learn without relying on replay buffers for reusing past off-policy data. We present an on-policy algorithm that trains Q-value models purely from on-policy trajectories, unlocking the possibility of using pathwise policy updates in the context of on-policy learning. We show how to combine stochastic policies for exploration with constrained updates for stable training, and evaluate important architectural components that stabilize value function learning. The result, Relative Entropy Pathwise Policy Optimization (REPPO), is an efficient on-policy algorithm that combines the stability of pathwise policy gradients with the simplicity and minimal memory footprint of standard on-policy learning. Compared to state-of-the-art on two standard GPU-parallelized benchmarks, REPPO provides strong empirical performance at superior sample efficiency, wall-clock time, memory footprint, and hyperparameter robustness.

1 INTRODUCTION

Most modern on-policy algorithms, such as TRPO (Schulman et al., 2015) or PPO (Schulman et al., 2017), use a score-based gradient estimator to update the policy. These methods have proven useful 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), but are often plagued by training instability. Zeroth-order, score-based gradient approximation exhibits high variance (Greensmith et al., 2004), which leads to unstable learning (Ilyas et al., 2020; Rahn et al., 2023), especially in high-dimensional continuous spaces (Li et al., 2018). In addition, it requires importance sampling to allow sample reuse, which exacerbates the high variance.

An alternative~~are pathwise policy gradient estimators (Silver et al., 2014), where a reparameterizeable policy class and a learned value function to approximate returns, are used to obtain a gradient estimate. This low-variance estimator directly optimizes the predicted returns and~~ , commonly used in off-policy learning, is to learn a parameterized state-action value function (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018), and use it to improve the policy, for example by using a pathwise policy gradient (Silver et al., 2014). Using a parameterized surrogate function to improve the policy often leads to faster ~~learning (Lillicrap et al., 2016). Furthermore, access to a state-action value estimate allows the agent to estimate the value of on-policy actions that were not executed in the environment. Therefore, we can forgo importance sampling , which greatly stabilizes multi-epoch training~~and more stable learning learning by reducing the score-based estimators variance (Mohamed et al., 2020) and by allowing us to remove importance sampling corrections.

However, the effectiveness of ~~pathwise policy gradients~~ these approaches is bounded by the quality of the approximate value function (Silver et al., 2014). As such, algorithms that use ~~pathwise policy gradients~~ a state-action value function usually rely on improving value learning through off-policy

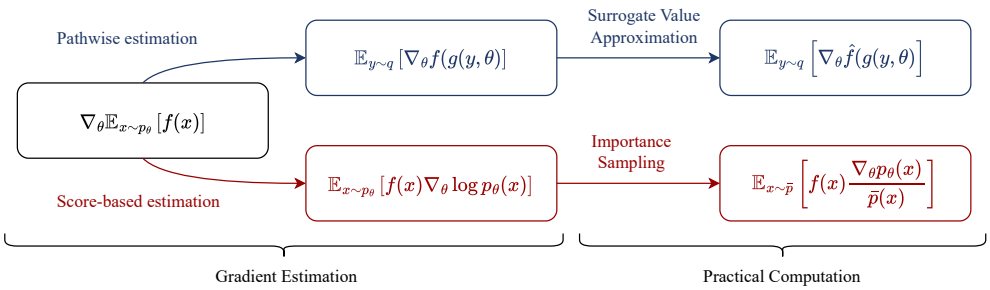


Figure 1: Overview of the strategies used by REPPO and PPO to obtain policy gradient estimators. Computing the gradient requires a mathematical transformation that allows for efficient estimation from samples, and additional steps that make the computation tractable in practice.

training (Fujimoto et al., 2018; Haarnoja et al., 2018). Unfortunately, off-policy training requires the use of replay buffers. Storing these replay buffers can be a challenge when the collected samples cannot fit in memory. In addition, training with past data introduces various 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 our core question:

Can we train a ~~robust~~ strong surrogate value function and effectively use ~~pathwise~~ it for policy ~~gradient~~ improvement in a fully on-policy setting without large replay buffers?

Building 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), we present an efficient on-policy algorithm, *Relative Entropy Pathwise Policy Optimization (REPPO)*, which uses the pathwise gradient estimator with an accurate surrogate value function learned from on-policy data. REPPO builds on the maximum entropy framework (Ziebart et al., 2008) to encourage exploration. It combines this with a KL regularization scheme, inspired by the Relative Entropy Policy Search method (Peters et al., 2010), which prevents aggressive policy updates from destabilizing the optimization.

Furthermore, we ~~incorporate several~~ evaluate several prominent advances in neural network architecture design to stabilize learning: categorical Q-learning (Farebrother et al., 2024), normalized neural network architectures (Nauman et al., 2024a; Hussing et al., 2024), and auxiliary tasks (Jaderberg et al., 2017). These components feature in many recent variants (Schwarzer et al., 2021; 2023; Nauman et al., 2024a; Hussing et al., 2024; Gallici et al., 2024; Lee et al., 2025a;b; Nauman et a of common value learning algorithm such as SAC (Haarnoja et al., 2018). We find that categorical Q-learning and normalization have a strong impact on the performance, while auxiliary tasks only show small impact, but become more relevant when reducing the amount of samples.

We test our approach in a variety of locomotion and manipulation environments from the Mujoco Playground (Zakka et al., 2025) and ManiSkill (Tao et al., 2025) benchmarks, and show that REPPO is competitive with tuned on-policy baselines in terms of sample efficiency and wall-clock time, 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. In introducing REPPO, our work makes the following contributions:

1. We showcase that ~~a~~ using a state-action value function and a pathwise policy gradient can be effective in on-policy RL~~.~~, as it allows on-policy action resampling, forgoing importance corrections. However, this requires learning a highly accurate state-action value function.
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.

We provide sample implementations in both the JAX (Bradbury et al., 2018) and PyTorch (Paszke et al., 2019) frameworks. Our code is available in the supplementary material of the submission.

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. We write $\mathcal{P}_\pi(x'|x)$ for the policy-conditioned transition kernel and $\mathcal{P}_\pi^n(y|x)$ for the n-step transition kernel. 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, $J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t r(x_t, a_t)\right]$, 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 $Q^\pi(x,a) = \mathbb{E}_\pi\left[\sum_{t=0}^\infty \gamma^t r(x_t,a_t)\middle| x_0 = x, a_0 = a\right]$. We use $\mu_\pi(y|x)$ to denote the discounted stationary distribution over states y when starting in state x. When $x \sim \mu_\pi(\cdot|y), y \sim \rho_0$, we will simply write $\mu_\pi(x)$ to denote the probability of a state under the discounted occupancy distribution.[1]

2.1 Policy Gradient Learning

~~Achieved returns (left) and path of four policies trained with different gradient estimation methods. We compare a score-function 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.~~

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

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{~~x\sim\mu_\pi}[~~Q^{\pi_\theta}(x,a)\nabla_\theta \log \pi_\theta(a|x)~~]_{x\sim\mu_\pi, a\sim\pi_\theta(\cdot|x)}[Q^{\pi_\theta}(x,a)\nabla_\theta \log \pi_\theta(a|x)]. \tag{1}$$

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.

An alternative approach, leveraged in off-policy learning, is the *deterministic policy gradient theorem* (DPG) (Silver et al., 2014). ~~To avoid confusion, as the DPG can also be used with stochastic policies (Haarnoja et al., 2018), we refer to it as the *pathwise policy gradient*, following Mohamed et al. (2020).~~ The estimator for the DPG relies on access to a differentiable state-action value function and a ~~reparametrizeable policy class~~[2] deterministic differentiable policy $\pi_\theta^{\det}(x)$. While access to the true value function is an unrealistic assumption, we can use a trained surrogate model, $\hat{Q}$, to obtain a biased estimate of the gradient

$$\nabla_\theta J(\pi_\theta) \approx \mathbb{E}_{~~x\sim\mu_\pi}[~~\nabla_a \hat{Q}^{\pi_\theta}(x,a)|_{a=\pi_\theta(x)} \nabla_\theta \pi_\theta(x)~~]_{x\sim\mu_\pi}[\nabla_a \hat{Q}^{\pi_\theta^{\det}}(x,a)|_{a=\pi_\theta^{\det}(x)} \nabla_\theta \pi_\theta^{\det}(x)]. \tag{2}$$

Finally, the DPG can be expanded to reparameterizeable stochastic policies[2]. We term this the *pathwise policy gradient*, following Mohamed et al. (2020), but the formulation has been used prominently in prior work such as SAC (Haarnoja et al., 2018), just without a proper name. The gradient estimator can be obtained from the following expectation

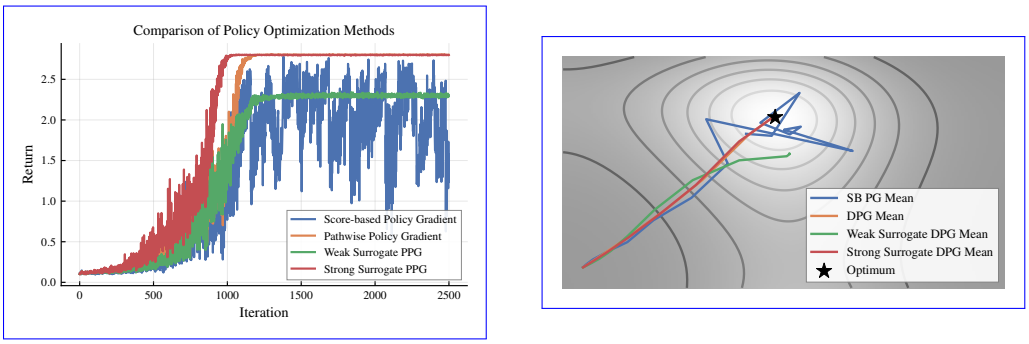
$$\nabla_\theta J(\pi_\theta) \approx \mathbb{E}_{x\sim\mu_\pi, \epsilon\sim p(\epsilon)}[\nabla_a \hat{Q}^{\pi_\theta^{\text{rep}}}(x,a)|_{a=\pi_\theta^{\text{rep}}(x,\epsilon)} \nabla_\theta \pi_\theta^{\text{rep}}(x,\epsilon)], \tag{3}$$

where $\pi_\theta^{\text{rep}}(x,\epsilon)$ is a reparameterization of $\pi_\theta(a|x)$. To avoid notational we will write $\pi_\theta(a|x)$ from now on to always mean the appropriate reparameterization.

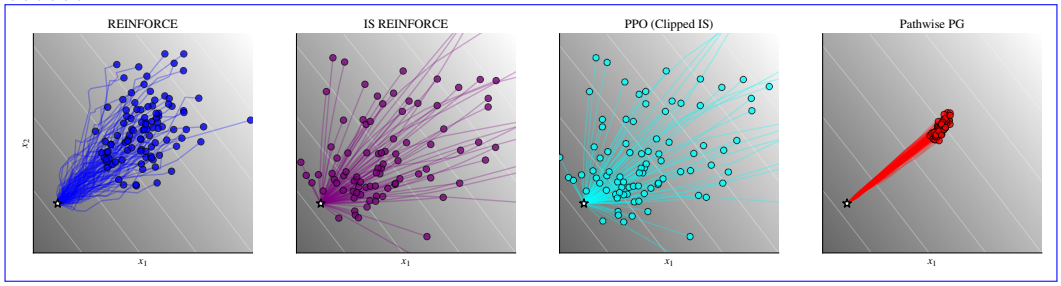
[1] A well-known issue of many policy gradient works is that in practice, they, perhaps erroneously, use the undiscounted empirical state occupancy for optimization (Nota & Thomas, 2020). REPPO similarly uses empirical samples without accounting for the discount factor in the objective.

[2] ~~We discuss an extension to non-reparametrizeable, discrete policies in .~~

[2] We discuss an extension to non-reparametrizeable, discrete policies in Appendix C.



(a) Achieved returns (left) and path of four policies trained with different gradient estimation methods. We compare a score-function 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.



(b) Gradient path over eight steps in the middle of the trajectory, visualized per algorithm for 8 steps. For Reinforce and PPG, new samples are drawn at every step. For the importance sampling based algorithms, one set of samples is sampled at the beginning and subsequent steps are conducted using importance sampling.

Figure 2: Visualization of gradient paths on a 2D example function.

2.2 ~~ILLUSTRATING~~ UNDERSTANDING SOURCES OF HARMFUL VARIANCE IN GRADIENT ESTIMATION

To build additional intuition on the differences between different policy gradient estimators, we conduct an illustrative experiment. Implementation details can be found in Appendix D.

On a simple objective $g(x)$ we initialize four Gaussians and update their parameters to maximize $J(\mu, \Sigma) = \mathbb{E}_{x\sim\mathcal{N}(\cdot|\mu,\Sigma)}[g(x)]$ with four different methods: a score-based policy gradient (using Equation 1), 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 3). We visualize the returns and the path of the mean estimates in Figure 2a. In addition, we zoom in on the gradient paths of the score-based estimator. We visualize 100 different eight step paths from the middle of the trajectory. Here, in addition to the vanilla score-based estimator, we also show an importance sampling and a clipped importance sampling estimator. These paths are visualized in Figure 2b.

The experiments shows that score-based gradient estimators have high variance, and can lead to unstable policies which fail to optimize the target. In addition, while importance sampling increases the sample efficiency of the algorithm, it greatly exacerbates these variance issues. We find that clipping the ratio estimate, as proposed by Schulman et al. (2017), prevents catastrophic instability, but does not reduce the variance substantially. On the other hand, using a pathwise ~~gradient~~ gradients is remarkably stable and exhibits small variance. However, it either requires access to the gradients of the objective function, or a strong surrogate model.

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

We now present our algorithm for using 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, as 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. Combining a set of recent advances in both reinforcement learning as well as neural network value function fitting, can satisfy these requirements. We first introduce the core RL algorithm, and then elaborate on the architectural design of the method.

At its core, REPPO proceeds similar to other on-policy actor-critic algorithms through three distinct phases: data gathering, value target estimation, and value and policy learning (see Algorithm 1). To obtain diverse data, REPPO uses a maximum-entropy formulation, adapted to multi-step TD-λ (Subsection 3.1), to encourage exploration. 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 and policy constraints (Subsection 3.2).

3.1 VALUE FUNCTION LEARNING

Off-policy PPG methods like TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018) mostly use single step Q learning, i.e. they use only immediate rewards for value function updates. This is paired with large replay buffers to stabilize learning. While on-policy algorithms cannot use past policy data, they can instead use low bias multi-step TD targets for stabilization (Fedus et al., 2020). Therefore, multi-step TD-λ targets form the basis for our value learning objective. Note that REPPO is more closely related to SARSA than to Q-learning (Sutton & Barto, 2018), due to being on-policy.

In addition to multi-step returns, 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 not only maximizes rewards but also penalizes the loss of entropy in the

policy distribution. The maximum-entropy policy objective (Levine, 2018) can be defined as

$$J_{\mathrm{ME}}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) + \alpha \mathcal{H}[\pi_\theta(x_t)]\right], \tag{4}$$

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

$$G^{(n)}(x_t, a_t) = \sum_{k=t}^{n-1} \gamma^k (r(x_k, a_k) - \alpha \log \pi(a_k|x_k)) + \gamma^n Q(x_n, a_n) \tag{5}$$

$$G^{\lambda}(x, a) = \frac{1}{\sum_{n=0}^{N} \lambda^n} \sum_{n=0}^{N} \lambda^n G^{(n)}(x, a), \tag{6}$$

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). Crucially, the targets are computed on-policy after a new data batch is gathered, and the Q targets are not recomputed before gathering new data. Our Q learning loss is

$$\mathcal{L}_Q^{\mathrm{REPPO}}\left(\phi | \{x_i, a_i\}_{i=1}^B\right) = \frac{1}{B}\sum_{i=1}^{B} \mathrm{HL}\left[Q_\phi(x_i, a_i), G^{\lambda}(x_i, a_i)\right] + \mathcal{L}_{\mathrm{aux}}(f_\phi(x_i, a_i), x_i'), \tag{7}$$

where x_i' refers to the next state sample starting from x_i, and HL is the HL-Gauss loss (see Subsection 3.3 and Subsection D.2), and $\mathcal{L}_{\mathrm{aux}}$ is presented in Subsection 3.3 and Subsection D.3.

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), so this simplification increases the robustness of our method. We also do not need a target value function copy, since we do not recompute the target at each step and it therefore remains on-policy.

3.2 Policy Learning

A core problem with value-based on-policy optimization is controlling the size of the policy update, as the value estimate is only accurate on the data covered by the prior policy. A large policy update can therefore destabilize learning (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 the behavioral (Peters et al., 2010; Schulman et al., 2015). To control the deviation, we use the Kullback-Leibler (KL) divergence, also called the relative entropy (Peters et al., 2010), as it can be justified theoretically through information geometry (Kakade, 2001; Peters & Schaal, 2008; Pajarinen et al., 2019), and is easy to approximate using samples.

Some works in the literature (Neumann, 2011; Sokota et al., 2022) claim that the reverse mode might be preferable for policy constraints, as it is mode-seeking, and the forward mode is mode-averaging. However, this intuition does not cleanly translate to our setting. As our policies are unimodal tanh-squashed Gaussian, the main impact of the KL direction is that the reverse-mode KL is entropy reducing. As we explicitly aim to increase the policy's entropy using the maximum entropy formulation, using forward-mode KL makes the optimization more stable.

Policy Optimization Objective Our policy updates derive from a constrained optimization problem which includes both entropy and the KL constraint, and where θ' is the behavior policy, and $\varepsilon_{\mathrm{KL}}$ and $\varepsilon_{\mathcal{H}}$ are the respective KL and entropy constraints

$$\max_{\theta} \quad \mathbb{E}_{\cancel{\rho_{\pi_{\theta'}}} x \sim \rho_{\pi_{\theta'}}}\left[\cancel{J_{\mathrm{ME}}} \mathbb{E}_{a \sim \pi_\theta(\cdot|x)}\left[Q(\cancel{\theta} x, a)\right]\right] \tag{8}$$

$$\text{subject to} \quad \mathbb{E}_{x \sim \rho_{\pi_{\theta'}}}\left[D_{\mathrm{KL}}\left(\pi_{\cancel{\theta}\theta'}(\cdot|x) \,\|\, \pi_{\cancel{\theta'}\theta}(\cdot|x)\right)\right] \leq \cancel{\mathrm{KL}_{\mathrm{tar}}} \varepsilon_{\mathrm{KL}} \tag{9}$$

$$\mathbb{E}_{x \sim \rho_{\pi_{\theta'}}}\left[\cancel{\mathcal{H}[\pi_\theta(x)]} \mathcal{H}[\pi_\theta(\cdot|x)]\right] \geq \cancel{\mathcal{H}_{\mathrm{tar}}} \varepsilon_{\mathcal{H}}. \tag{10}$$

A similar combination of maximum entropy and KL divergence bound has been explored in various forms (Abdolmaleki et al., 2015; Pajarinen et al., 2019; Akrour et al., 2019). However, while previous approaches use complex solutions to this problem, such as approximate mirror descent, line search, or heuristic clipping, we take a simpler approach. We relax the problem, which introduces two hyperparameters, α for the entropy, and β for the KL. Inspired by Haarnoja et al. (2019), REPPO automatically adapts these constraints when the policy violates them.

Policy Updates and Multiplier Tuning In the constrained objective, we introduce two hyperparameters, ~~$\mathcal{H}_{\text{tar}}$ and KL_{tar}~~ $\varepsilon_{\mathcal{H}}$ and ε_{KL}, which bound the entropy and KL divergence. The goal of the 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

$$\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha e^\alpha \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}} \left[(\mathcal{H}[\pi_\theta(x)] \mathcal{H}[\pi_\theta(\cdot|x)] - \mathcal{H}_{\text{tar}} \varepsilon_{\mathcal{H}}) \right] \tag{11}$$

$$\beta \leftarrow \beta - \eta_\beta \nabla_\beta e^\beta \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}} \left[(\text{D}_{\text{KL}}(\pi_{\theta\theta'}(\cdot|x) \| \pi_{\theta'\theta}(\cdot|x)) - \text{KL}_{\text{tar}} \varepsilon_{\text{KL}}) \right]. \tag{12}$$

Finally, to ensure our KL constraint ~~it~~ is (approximately) maintained, we clip the actor loss based on whether the constrained is currently violated. The full policy objective for REPPO is now

$$\mathcal{L}_\pi^{\text{REPPO}}(\theta|\{x_i\}_{i=1}^B) = \frac{1}{B}\sum_{i=1}^B \begin{cases} -Q(x_i,a) + e^\alpha \log \pi_\theta(a|x_i), & \text{if } \frac{1}{k}\sum_{j=1}^k \log \frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} < \varepsilon_{\text{KL}} \\ e^\beta \frac{1}{k}\sum_{j=1}^k \log \frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)}, & \text{otherwise} \end{cases} \tag{13}$$

where a is sampled from $\pi_\theta(\cdot|x_i)$ and a_j from the past behavior policy $\pi_{\theta'}(\cdot|x_i)$~~.~~, and k denotes how many samples are used to approximate the KL. As with the critic, the optimized loss is a mean over a minibatch from the rollout data. Note that contrary to other on-policy algorithms like PPO and TRPO, we are not forced to use actions sampled from the behavior policy in the policy gradient estimator, which removes the need for importance sampling correction. We will show that this greatly improves the performance of REPPO in Subsection 4.1.

Jointly tuning the entropy and KL multipliers is a crucial component of REPPO. As the policy entropy and KL are tied, letting the entropy of the behavior policy collapse results in a scenario where 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, 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.

3.3 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.

Cross-entropy loss for regression The first choice is to replace 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), which can 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. Palenicek et al. (2025) further investigate and reinforce this claim, showing that stable gradients arise from cross-entropy based losses. We present the mathematical form of the loss formulation in Subsection D.2. We find that a categorical loss is a crucial addition, as our ablation experiments show (Subsection E.1), but alternatives like C51 could easily work as well.

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 shown the importance of layer

normalization (Ba et al., 2016) for stable critic 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 off-policy bootstrapped methods; yet, we still see performance benefits in most environments from normalization.

Auxiliary tasks Auxiliary tasks (Jaderberg et al., 2017) can stabilize features in environments with sparse rewards, where the lack of a reward signal can prevent learning meaningful representations via the Q learning objective (Voelcker et al., 2024a). For REPPO, auxiliary tasks are especially impactful when we decrease the number of samples used in each update batch (see Subsection E.1). We provide a discussion of this auxiliary task setup, including the loss function, in Subsection D.3.

4 Experimental Evaluation

We begin by evaluating whether pathwise estimators improve upon score-based estimation in on-policy RL settings. We then compare our approach to baselines, evaluating final performance, sample and wall-clock efficiency, and stability of policy improvement. Our results demonstrate strong performance of REPPO on all axes. Additional details on architectures, hyperparameters, and ablations are provided in Subsection D.4 and Appendix E. A discrete variant of REPPO, along with its architectural changes and experimental results, is presented in Appendix C.

Environments We evaluate REPPO on two major GPU-parallelized benchmark suites: 23 tasks from the mujoco_playground DMC suite (Zakka et al., 2025) and 8 ManiSkill environments (Tao et al., 2025), covering locomotion and manipulation, respectively. These tasks span high-dimensional control, sparse rewards, and chaotic dynamics.

4.1 Score-based and Pathwise Comparison

REPPO offers an alternative to score-based policy gradient estimation in on-policy RL. However, we also introduce several enhancements, including automated tuning of entropy and KL coefficients, to improve value and policy learning. To assess the benefits of learned values and pathwise gradient estimation over score-based methods, we conduct two experiments. First, we replace the pathwise term $-Q(x, a)$ in Equation 13 with the score function $\log \pi(a|x)[Q(x, a)]_{\text{sg}}$, denoted as *REPPO (score-based, Q)*. Second, we replace the gradient estimator with the GAE-based clipped objective from PPO, denoted as *REPPO (score-based, GAE)*. Aggregate results are presented in Figure 3.

Using the approximate Q function in the policy gradient objective provides a strong improvement over PPO or REPPO with a clipped objective. Q score-based REPPO outperforms PPO, ~~clarifying~~ strongly showcasing the benefits of value function learning ~~. This further~~ and removing importance sampling. This also shows that the REPPO framework can ~~also~~ be used with policy classes that are not amenable to reparameterization, such as diffusion policies (Chi et al., 2024; Celik et al., 2025; Ma et al., 2025), by using a score-based estimator together with the learned Q function. Interestingly, combining the PPO objective with REPPO leads to slightly worse results than vanilla PPO. We find that the high variance complicates the automatic parameter tuning scheme.

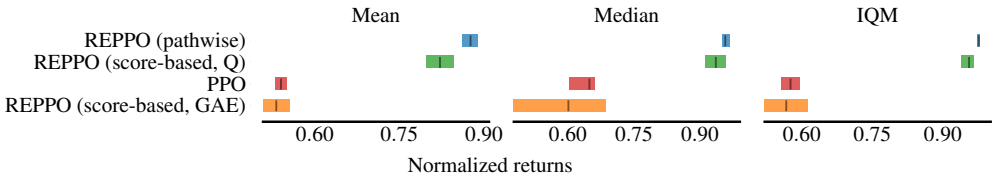
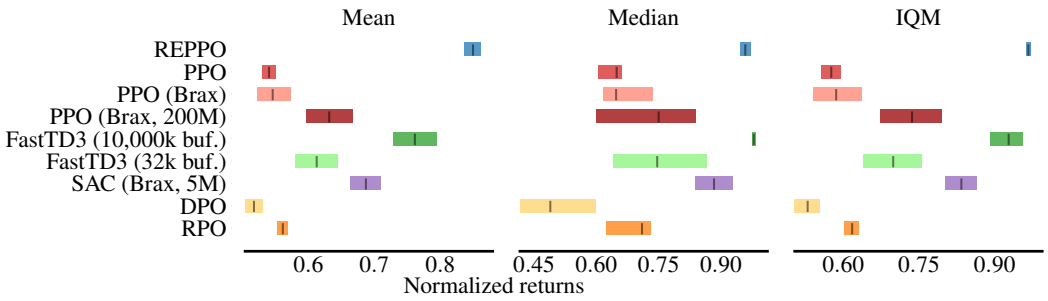
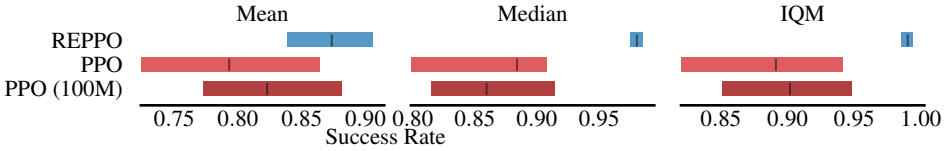


Figure 3: Aggregate performance metrics on the mujoco_playground benchmark. We compare REPPO with two ablations: one using the score-based gradient estimator with the learned Q function, and another using an on-policy GAE estimate with importance sampling and clipping. For additional context, we also report PPO results.



(a) Aggregate performance metrics on the mujoco_playground DeepMind Control Suite benchmark. We compare both REPPO and our 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), as well as FastTD3 (Seo et al., 2025), RPO (Rahman & Xue, 2023), and DPO (Lu et al., 2022).



(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 than 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 4: Aggregate performance comparison on (a) mujoco_playground DMC and (b) ManiSkill3.

4.2 Benchmark comparison

We compare REPPO against the PPO and SAC results reported by Zakka et al. (2025) and Tao et al. (2025). We report PPO baselines at 50M environment steps, and at the larger training horizon used in the original papers (Zakka et al., 2025). Results taken from Zakka et al. (2025) are denoted as "PPO/SAC (Brax)". To ensure that PPO is not undertuned for the 50m step regime we re-tuned the hyperparameters of the implementation provided by Lu et al. (2022). SAC results are reported at 5m steps as this amounts to similar total runtime as the 200m PPO results (compare results in Zakka et al. (2025). Naively running SAC at a larger sample budget and wall-clock efficiency can lead to instability, as Seo et al. (2025) demonstrates. Furthermore, we include FastTD3 (Seo et al., 2025) on DMC locomotion tasks, trained under two memory budgets: the default replay buffer (10,485,760 transitions) and a constrained buffer similar in size to on-policy methods (32,768 transitions) to control for the the memory and performance trade-off. Finally, we compare against Robust Policy Optimization (RPO) (Rahman & Xue, 2023) and Discovered Policy Optimization (DPO) (Lu et al., 2022). However, even with some hyperparameter tuning, we were unable to ~~get performamnce above~~ achieve a strong performance improvement beyond the PPO baseline with these ~~appraoches~~approaches.

For REPPO, we report results aggregated over 20 seeds across all tasks. We run 20 seeds for PPO and 5 for FastTD3[3], reporting aggregate scores with 95% bootstrapped confidence intervals (Agarwal et al., 2021). To enable aggregation across tasks, returns on mujoco_playground are normalized by the maximum achieved by any algorithm, while for ManiSkill we report raw success rates, which are naturally comparable across tasks.

Final Performance and Sample Efficiency We first investigate the performance of policies trained using REPPO. We report aggregate performance at the end of training on both benchmarks in Figure 4. For both benchmarks, we also provide the corresponding training curves in Figure 5.

[3]We use fewer seeds for FastTD3 as we are unable to replicate the speed claimed in the paper. This is due pytorch specific issues discussed in Appendix B, and because we use smaller GPUs for our experiments.

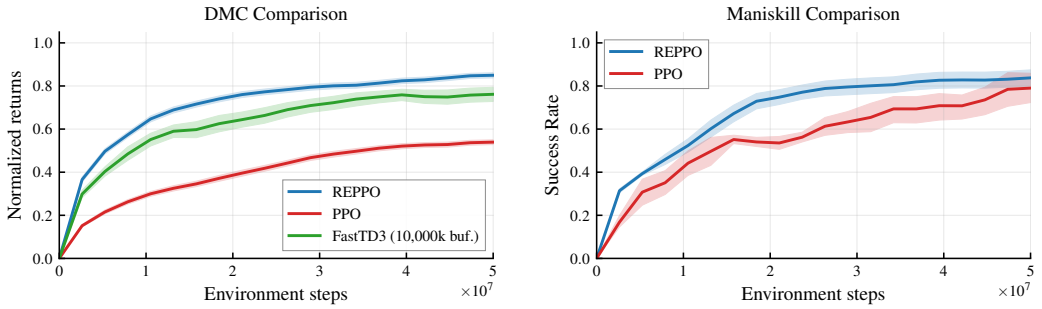


Figure 5: Aggregate sample efficiency curves for the benchmark environments. Settings are identical to those in Figure 4. REPPO achieves higher performance at a faster rate in both benchmarks.

The aggregate results shown in Figure 4 and Figure 5 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 (Chan et al., 2020a; Agarwal et al., 2021).

We find that PPO struggles on high-dimensional tasks such as HumanoidRun, even with large batch sizes aimed at reducing policy gradient variance. Moreover, despite its approximate trust-region updates, PPO suffers from performance drops and unstable training. This erratic behavior closely mirrors the score-based policy gradient instability shown in Figure 2a. In contrast, REPPO exhibits more stable improvements and lower variance across seeds.

Wall-clock Time Wall-clock time is an important metric~~in simulation~~, as it reflects the practical utility of an algorithm: faster training enables more efficient hyperparameter search and experimentation. However, measuring wall-clock time is nuanced, as results heavily depend on implementation details and are difficult to reproduce. We discuss these challenges across different frameworks in Appendix B. In Figure 6, we compare the wall-clock performance of our approach against PPO and SAC in JAX. Other baselines lack JIT-compilable implementations, making direct comparisons less meaningful.

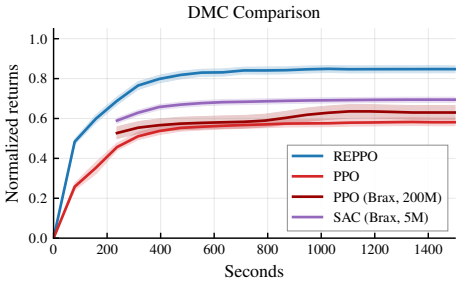


Figure 6: Wall-clock time comparison of REPPO against ~~common~~ PPO and SAC implementations in JAX. REPPO matches ~~PPO~~other algorithms' ~~s~~ speed but achieves higher return.

The computational cost per update is higher for REPPO than for PPO due to larger default networks and gradient propagation through the critic–actor chain. Nevertheless, both algorithms converge on most tasks in roughly 600–800 seconds, with REPPO achieving about 33% higher normalized returns. This shows that the sample efficiency of pathwise gradients can offset their higher per-update cost, yielding improved wall-clock efficiency compared to score-based PPO. In addition, we find that jax-based SAC, which is tuned to trade sample for computational efficiency, slightly outperforms PPO, but does not match REPPO in performance. We note that other, modern SAC implementations (Nauman et al., 2024b; Lee et al., 2025a;b), are able to achieve better performance, but at the cost of computational efficiency.

Reliable Policy Success We further investigate the stability of policy improvements using score-based and pathwise policy gradients. Our guiding principle is that such updates should not cause large drops in performance. To capture this, we adopt the "reliable success" metric, as proposed in Chan et al. (2020b). We define an algorithm as *reliably performant* if, once its performance exceeds a fixed threshold τ, it never drops below this threshold thereafter. At each timestep, we track the number of runs that satisfy this criterion. This metric reflects the practical requirement that

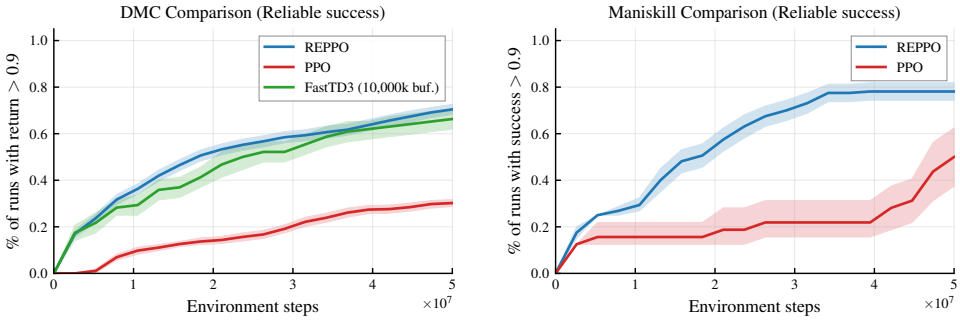


Figure 7: Fraction of runs that achieve reliable performance as measured by our metric for policy stability and reliability. REPPO's immediately starts achieving high performance in some runs and the number gradually increases indicating stable learning. PPO struggles to achieve high performance initially and to maintain high performance throughout training.

a deployed algorithm should not suddenly degrade simply due to continued training. We report the percentage of reliably successful runs for both REPPO and PPO in Figure 7.

On both DMC and ManiSkill benchmarks, REPPO achieves reliable performance improvements quickly, with success rates and returns steadily increasing. By the end of training, about four out of five runs have reached the threshold of $\tau = 0.9$ without dropping below it, whereas PPO achieves roughly 40 percentage points fewer reliably performant runs. We also find notable differences in sample efficiency: PPO requires 5–10 million interactions before most envs become reliably performant. Overall, these results show that, despite relying on a biased surrogate value model, pathwise policy gradients enable stable long-term improvement.

5 Conclusion and Avenues for Future Work

In this paper we present REPPO, a highly performant ~~but~~ yet efficient on-policy algorithm that leverages ~~pathwise instead of score-based~~ trained state-action value functions and pathwise 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.

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. We also observe that removing the importance sampling step in PPO has a crucial impact on performance, which suggests further research on the trade-off between efficiency and stability in on-policy gradient estimation is needed. 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.

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 *Proceedings of the 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 *Advances in Neural Information Processing Systems*, 2022.

Riad Akrour, Joni Pajarinen, Jan Peters, and Gerhard Neumann. Projections for approximate policy iteration algorithms. In *Proceedings of the 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.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. DIME: Diffusion-based maximum entropy reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2025.

Stephanie Chan, Sam Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *Proceedings of the International Conference on Learning Representations*, 2020a.

Stephanie CY Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *Proceedings of the International Conference on Learning Representations*, 2020b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

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.

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 *Proceedings of the 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 *Proceedings of the International Conference on Learning Representations*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the International Conference on Machine Learning*, 2023.

Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In *Proceedings of the 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 *Proceedings of the 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. *arXiv preprint arXiv:1812.05905*, 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 *Proceedings of the International Conference on Learning Representations*, 2024.

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 *Proceedings of the International Conference on Learning Representations*, 2020.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *Proceedings of the 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.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*, 2017.

Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip Thomas. Evaluating the performance of reinforcement learning algorithms. In *Proceedings of the International Conference on Machine Learning*, 2020.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 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 *Proceedings of the 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 *Proceedings of the 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 International Joint Conference on Artificial Intelligence*, 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*, 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. In *Proceedings of the Conference on Lifelong Learning Agents*, 2024.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Proceedings of the International Conference on Machine Learning*, 2025.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*, 2017.

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.

Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. No representation, no trust: Connecting representation, collapse, and trust issues in PPO. In *Advances in 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 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.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=zhOUfuOIzA.

Gerhard Neumann. Variational inference for policy search in changing situations. In *Proceedings of the International Conference on International Conference on Machine Learning*, 2011.

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.

Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

Fabian Otto, Philipp Becker, Vien Anh Ngo, Hanna Carolin Maria Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2021.

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, 2022.

Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible natural gradient policy search. *Machine Learning*, 108(8), 2019.

Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. *arXiv preprint arXiv:2509.25174*, 2025.

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*, 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 AAAI Conference on Artificial Intelligence*, 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.

Md Masudur Rahman and Yexiang Xue. Robust policy optimization in deep reinforcement learning, 2023.

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, 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. *arXiv preprint arXiv:2505.22642*, 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.

Samuel Sokota, Ryan D'Orazio, J Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, Noam Brown, and Christian Kroer. A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Sanghyun Son, Laura Yu Zheng, Ryan Sullivan, Yi-Ling Qiao, and Ming Lin. Gradient informed proximal policy optimization. In *Advances in Neural Information Processing Systems*, 2023.

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 *Proceedings of the International conference on Learning Representations*, 2019.

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*, 2022.

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, 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 *Proceedings of the 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.

Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Proceedings of the 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 *Proceedings of the International Conference on Learning Representations*, 2022.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving q-value divergence in offline-RL. In *Advances in 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 National Conference on Artificial Intelligence*, 2008.

A Extended 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).

Incorporating ground-truth gradient signal to stabilize training has also 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., 2022).

Trust regions and constrained policy optimization Other approaches have used similar KL and trust region constraint as REPPO. 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). Akrour et al. (2019) propose to project the policy onto the trust-region to sidestep the difficulty associated with clipping. We find that our approach is simpler to implement and more general, as we do not assume direct projection is possible.

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.

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. We also evaluate our approach on discrete action spaces (see Appendix C). While PQN performs slightly better, likely owing to tuned exploration techniques, we show that our method works robustly across both discrete *and* continuous action spaces.

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.

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 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. This is the reason why we do not include FastTD3 in Figure 6, as only a PyTorch implementation of the algorithm exists. FastTD3 and REPPO use similar algorithms and hyperparameters, therefore, barring complexities like those discussed below, we expect them to perform at similar speeds.

More importantly, torch's compilation libraries are built to accelerate standard supervised and generative workflows, but do not support RL primitives equally well. As the CPU needs to load kernels during training which the GPU then executes, the CPU plays a much larger role in the speed measurements of the torch-based variant of REPPO. Especially the tanh-squashed log probability computation and the frequent resampling from the action space cannot be offloaded into an efficient

kernel without providing one manually, which we have not done. This is likely due to the fact that torch keeps its random seed on the CPU. This is not a concern for jax, due to the fact that all kernels are statically compiled when the program is first executed, and random seeds are handled explicitly as part of the program state. Therefore, 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.

C DISCRETE REPPO (D-REPPO)

One of the major advantages of PPO in the zoo of RL algorithms is the fact that it can be used in both continuous and discrete action settings. However, as we build on the DDPG/TD3/SAC line of work, the exposition of our algorithm has focused on the continuous setting alone.

Nonetheless, it is easy to adapt our approach to the discrete action setting as well. Following the proposal of Christodoulou (2019), we can circumvent the chained critic-actor gradient and compute the value of the current policy, the entropy, and the KL bound in closed form

$$\mathcal{L}^{\mathrm{D-REPPO}}_{\pi,\leq\mathrm{KL}}(\theta|B) = -\frac{1}{|B|}\sum_{i=1}^{|B|}\sum_{j=1}^{|A|}\pi_\theta(a_j|x_i)\left(Q(x_i,a_j) + e^\alpha \log \pi_\theta(a_j|x_i)\right) \tag{14}$$

$$\mathcal{L}^{\mathrm{D-REPPO}}_{\pi,>\mathrm{KL}}(\theta|B) = -\frac{1}{|B|}\sum_{i=1}^{|B|} e^\beta \sum_{j=1}^{|A|}\pi_{\theta'}(a_j|x_i)\log\frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} \tag{15}$$

$$\mathcal{L}^{\mathrm{D-REPPO}}_{\pi}(\theta|B) = \begin{cases}\mathcal{L}^{\mathrm{D-REPPO}}_{\pi,\leq\mathrm{KL}}(\theta|B), & \text{if } \sum_{j=1}^{k}\log\frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} < \varepsilon_{\mathrm{KL}} \\ \mathcal{L}^{\mathrm{D-REPPO}}_{\pi,>\mathrm{KL}}(\theta|B), & \text{otherwise.}\end{cases} \tag{16}$$

This variant of our algorithm still directly differentiates the full Q function objective, so can still be seen as a pathwise implementation. But computing the expectation in closed form circumvents the necessity to use a biased estimator for discrete sampling, such as the Gumbel-Softmax trick (Maddison et al., 2017; Jang et al., 2017; Fujimoto et al., 2024).

To investigate the benefits of our approach in the discrete action setting, we compare it against PQN (Gallici et al., 2024) and PPO. The main benefit of our approach over PQN is that it is a) a general algorithm that unifies both discrete and continuous action spaces, due to the underlying actor critic architecture, and b) that the principled entropy and KL objectives stabilize updates and encourages continuing exploration without an epsilon greedy exploration strategy.

We find that our algorithm is able to perform roughly on-par with PQN in the Atari-10 suite of games (cf. Table 1 and Figure 8) with only minor changes to the architecture to adapt to the Atari games benchmark. Notably, suitable settings for the KL and entropy target remain consistent even for the discrete action setting. We only find that the value of $\lambda = 0.65$ that is also recommended by Gallici et al. (2024) is superior to our default value of 0.95, likely due to the higher variance of the return in the atari games. While the high variance across Atari games makes drawing a clear conclusion difficult, we find that PQN seems to achieve slightly better performance. We find that this is most likely due to the fact that the algorithm adds explicit exploration noise, while we rely on the entropy and conservative KL terms to pace policy improvement.

Table 1: Aggregated Human-Normalized Atari-10 scores with 95% confidence intervals.

Algorithm	Mean [CI]	Median [CI]	IQM [CI]
REPPO	2.98 [2.64, 3.33]	1.68 [1.48, 1.82]	1.64 [1.54, 1.74]
PQN	3.35 [3.00, 3.76]	1.58 [1.48, 1.71]	1.64 [1.58, 1.71]

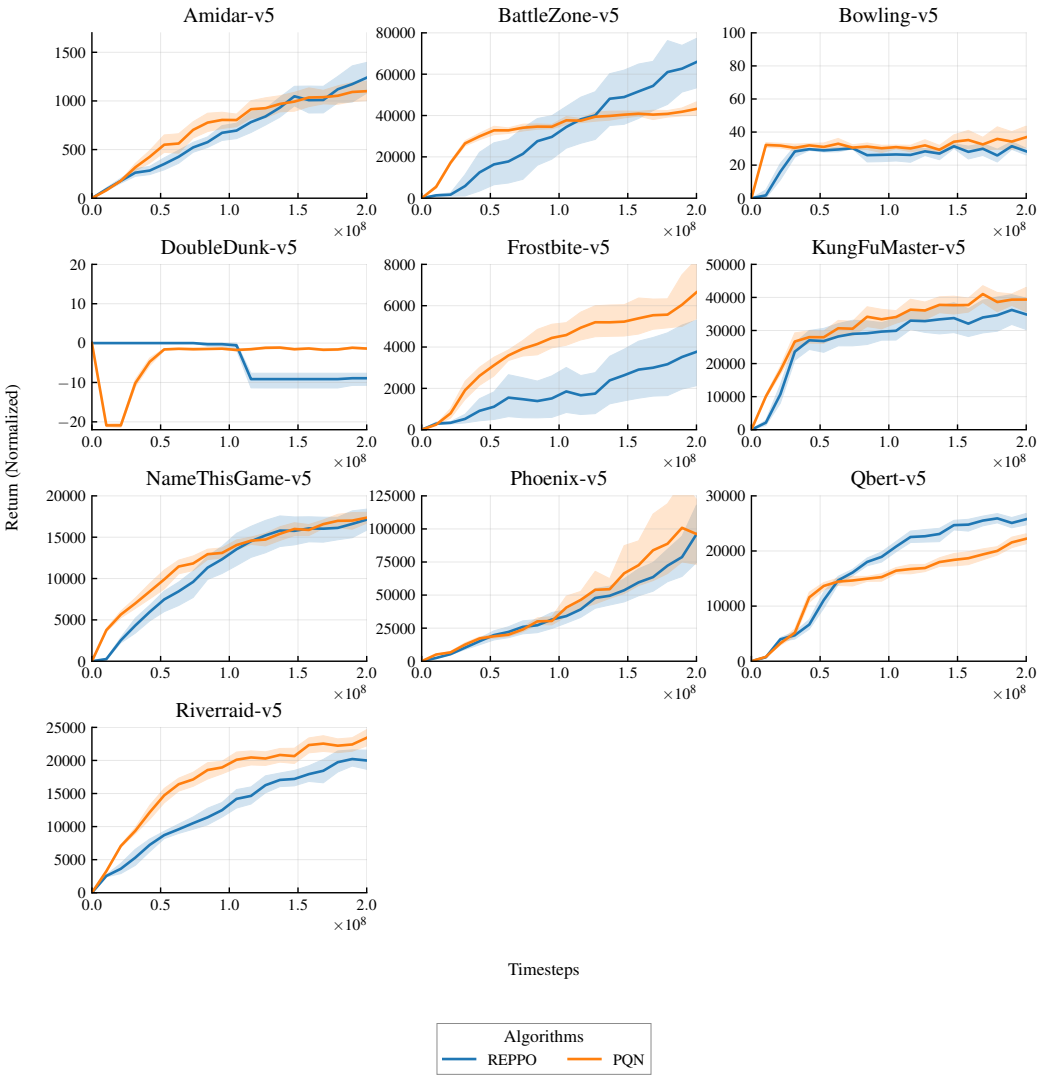


Figure 8: Per-environment results on the Atari-10 suite

D Implementation details and hyperparameters

In the following, we present implementation details on experiments, as well as a hyperparameter overview.

D.1 Toy example

To obtain the gradient descent comparison in Subsection 2.2 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

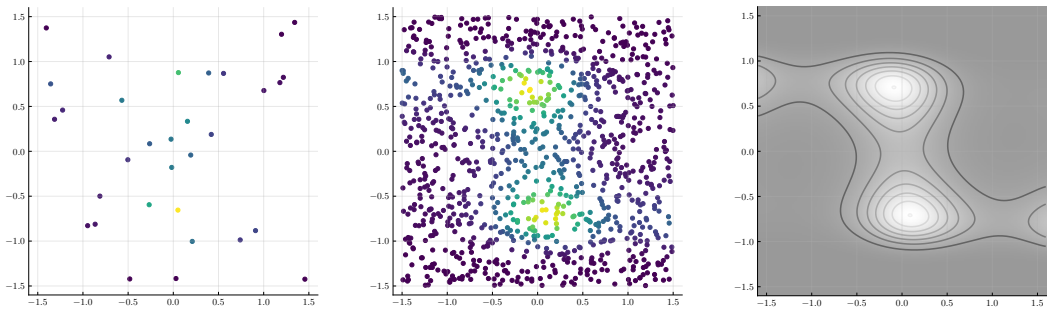


Figure 9: Samples used to train the surrogate function. On the left, we visualize the 32 sample dataset to train the weak surrogate function, in the middle the 1024 datapoints to train the strong, and on the right the full objective function.

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.2 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

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

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 $\max$.

D.3 Auxiliary Task Setup

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

$$\mathcal{L}_{\mathrm{aux}}(x_t, a_t, x_{t+1}, a_{t+1}) = \left| f_p(\phi(x_t, a_t)) - \phi(x_{t+1}, a_{t+1}) \right|^2 . \tag{17}$$

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

Environment	
total time steps	$50,000,000$
n envs	1024
n steps	128
$\mathrm{KL}_{\mathrm{tar}}$	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	$1 - \frac{10}{\text{max env steps}}$
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
~~$\mathrm{KL}_{\mathrm{tar}}$~~ $\varepsilon_{\mathrm{KL}}$	0.1
α start	0.01
~~$\mathcal{H}_{\mathrm{tar}}$~~ $\varepsilon_{\mathcal{H}}$	$0.5 \times \dim \mathcal{A}$
aux loss mult	1.0

Table 2: Default REPPO hyperparameters

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.

D.4 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 2. 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). This makes the hyperparameters, together with the algorithm description, and the source code, a *complete algorithm specification* in the sense of Jordan et al. (2020), as we only vary hyperparameters across environments following simple equations on clear, domain sepcific hyperparameters such as the size of the action space and the length of the experiment.

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.

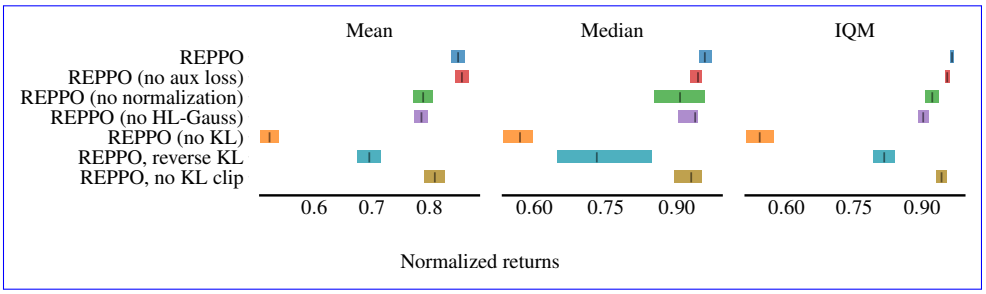
For more exact details on the architecture we refer to interested readers to the codebase.

E Additional Results

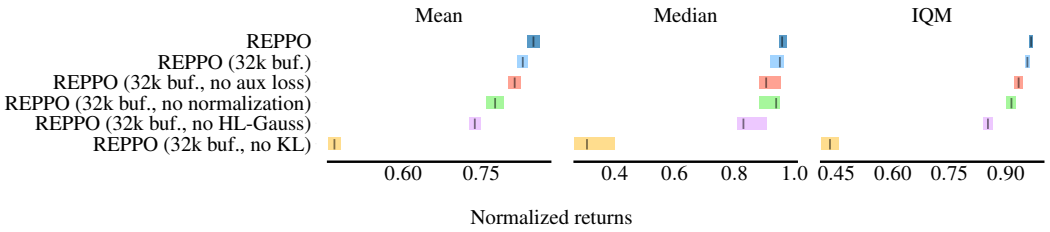
In the following, we provide additional results and further clarification on existing experiments in Section 4.

E.1 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



(a) Large dataset size ablation (128×1024).



(b) Small dataset size ablation (32×1024).

Figure 10: Ablation on components and data size on the DMC benchmark. 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.

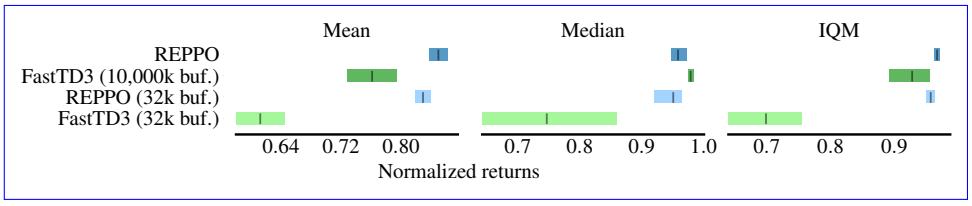


Figure 11: Comparison of aggregate performance between REPPO and FastTD3. REPPO is competitive with the large buffer FastTD3 version and outperforms FastTD3 when memory is limited.

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 $32,768$.

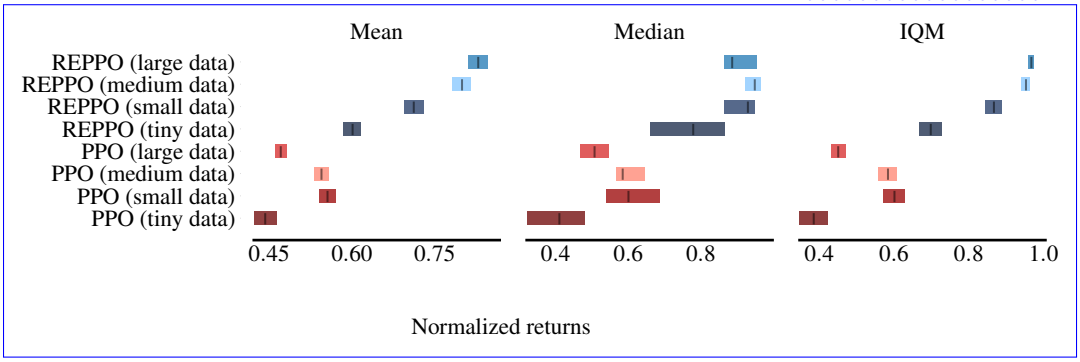
As shown in Figure 10, our results indicate that both the KL regularization of the policy updates and the categorical Q-learning via HL-Gauss are necessary to achieve strong performance independent of the size of the on-policy data used to update our model. We find that the KL divergence is the only 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 a more clearly negative 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.

E.2 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

	Num envs	Num steps	Num minibatches	Epochs	Updates per batch
Large data	1024	128	64	8	512
Medium data	1024	32	16	8	128
Small data	1024	8	4	8	32
Tiny data	256	8	1	8	8

(a) ~~Comparison of aggregate performance between and FastTD3. is competitive with~~ Dataset configurations for the ~~large buffer FastTD3 version and outperforms FastTD3 when memory is limited~~data scaling experiment.



(b) Aggregated performance of REPPO and PPO under different batch dataset sizes. The mean performance of REPPO drops monotonically with decreasing batch size, while PPO shows its highest performance with a medium and small dataset size.

Figure 12: Experiment to compare the impact of batch datset size on different on-policy algorithms.

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 FastTD3 (Seo et al., 2025) which also uses GPU-parallelized environments but operates off-policy. 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 11.

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.

E.3 Data scaling

To further understand what enables REPPO to perform well, we take a detailed look at the interplay between batch size and gradient steps. In our default configuration, REPPO uses very long rollouts and a high number of parallel environments, as well as a large number of policy and value function update steps. PPO on the other hand works best at smaller dataset sizes. We therefore set up REPPO and PPO training runs across 4 datasets, varying the rollout length. To keep the total number of gradient steps and the minibatch size the same, we reduced the number of minibatches proportionally to the batch size. The settings are summarized in Figure 12a. Note that in the large settings, the data becomes more off-policy. Both PPO and REPPO have explicit ways to deal with this, clipping and the KL minimization term respectively, but the clipping term in PPO is only a heuristic to prevent large importance sampling ratios.

Comparing the performance of both approaches (see Figure 12b), we observe a clear pattern. The mean performance of REPPO drops steeply with decreasing dataset size. PPO on the other hand does best in the medium and small dataset regimes. This highlights the different mechanisms on which both algorithms operate. Larger datasets allow the trained Q function to generalize better, similar to the insight presented in Figure 2a. On the other hand, for PPO the dataset size needs to be

large enough to allow for stable gradient estimation, but not so large that too many gradient update steps are necessary. This is because clipping can prevent further learning, and many update steps can exacerbate varaince issues with importance sampling.

Note that at some point, REPPO will likely also stop improving with larger datasets and more gradient update steps. We see that the performance differences between the medium and the large dataset at not as strong as with smaller datasets. REPPO cannot continue to learn on fixed data forever, by design, as the KL divergence between two consecutive policies is constrained. However, we can hypothesize based on the empirical evidence that REPPO is able to scale more gracefully with large amounts of data.

E.4 Per Environment Sample Efficiency Curves

Finally, we provide sample efficiency curves per environment in Figure 13, Figure 14, and Figure 15.

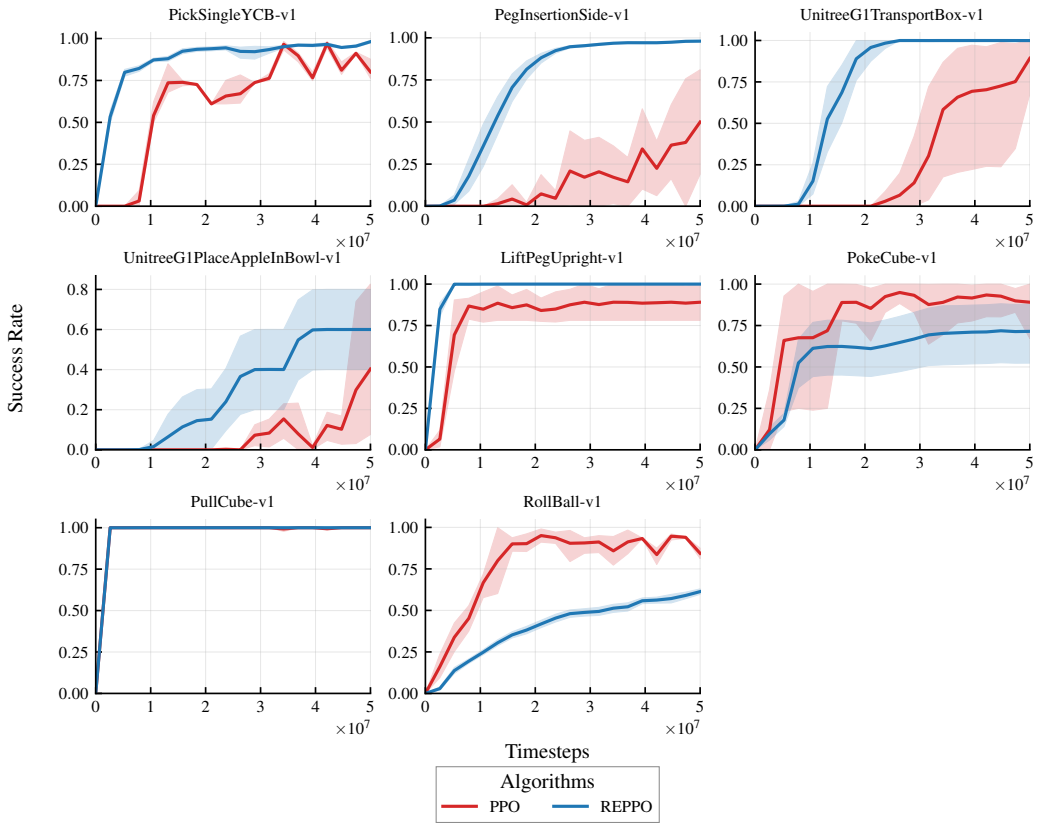


Figure 13: Per-environment results on the ManiSkill suite

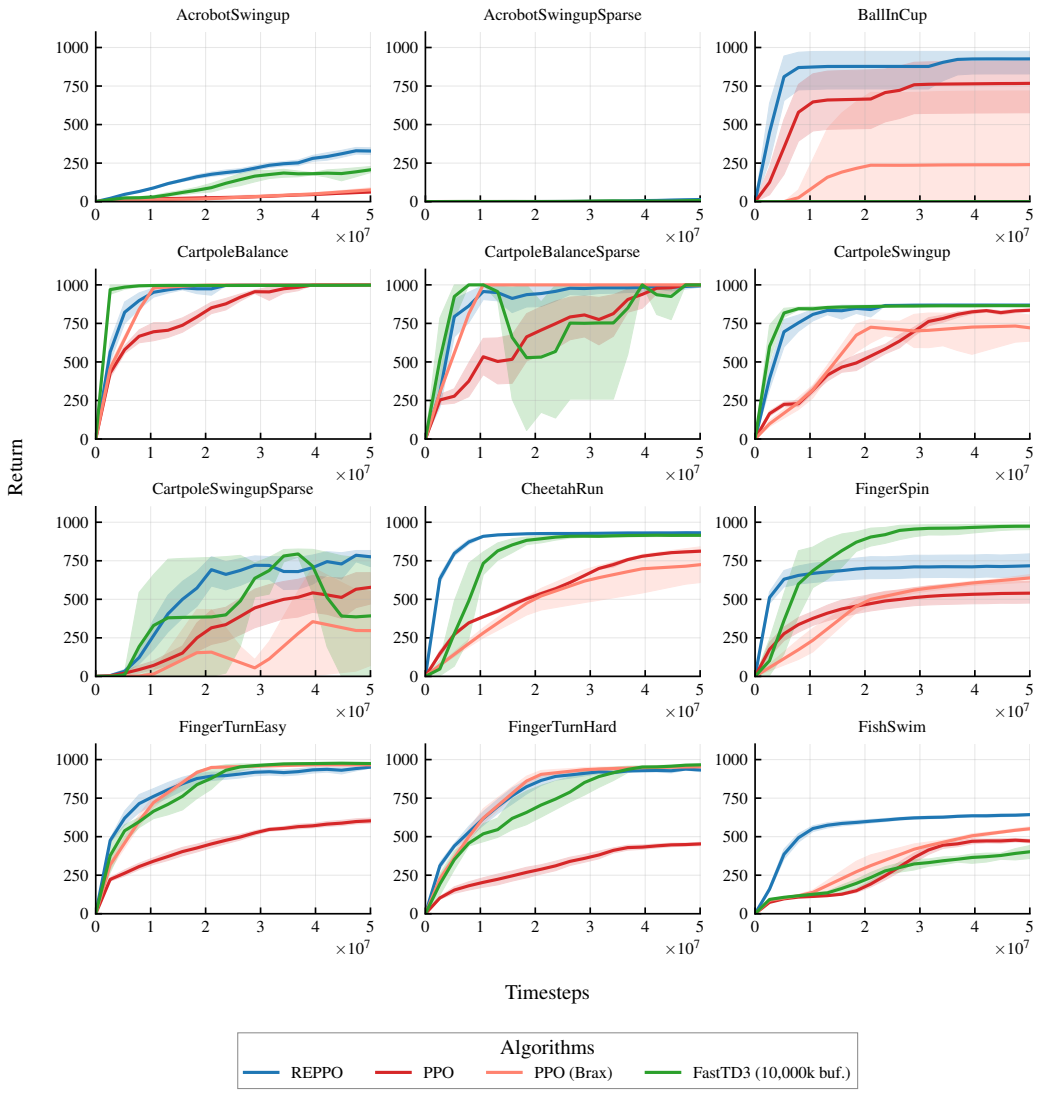


Figure 14: Per-environment results on the mujoco_playground DMC suite

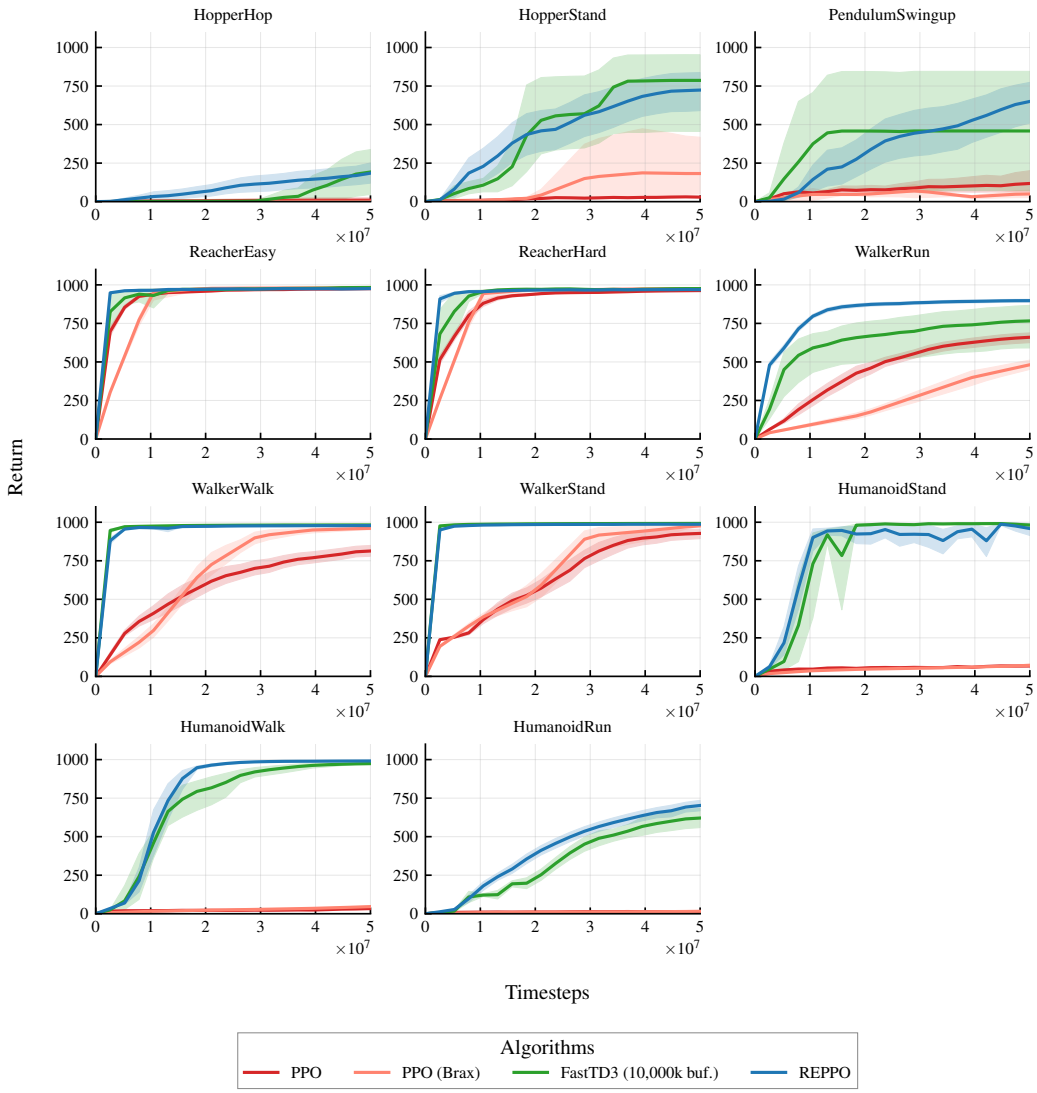


Figure 15: Per-environment results on the mujoco_playground DMC suite

F PSEUDOCODE

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 approximate $V_{t+1} \leftarrow Q_\phi(x_{t+1}, a_{t+1})$ with $a_{t+1} \sim \pi_{\theta'}(\cdot|x_{t+1})$
    Compute $\psi_t \leftarrow f_\phi(x_{t+1}, a_{t+1})$
    // Maximum entropy augmented reward, see Subsection 3.1
    $\tilde{r}_t \leftarrow r_t - \alpha \log \pi_\theta(a_{t+1}|x_{t+1})$
    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 Subsection 3.1
  for $t = T - 1$ *down to* 0 **do**
    $G_t^\lambda \leftarrow \tilde{r}_t + \gamma[(1 - d_t)(\lambda G_{t+1}^\lambda + (1 - \lambda)V_{t+1})]$
  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 Subsection 3.3
      $L_Q \leftarrow \frac{1}{B} \sum \text{CrossEntropy}(Q_\phi(x_i, a_i), \text{Cat}(G_i^\lambda))$
      // Auxiliary task, see Subsection 3.3
      $L_{aux} \leftarrow \frac{1}{B} \sum ||f_\phi(x_i, a_i) - \psi_i||^2]$
      Update critic: $\phi \leftarrow \phi - \alpha_Q \nabla_\phi (L_Q + \beta L_{aux})$
      // Actor update with entropy and KL regularization, see Subsection 3.1 and Subsection 3.2
      Sample action $a_i' \sim \pi_\theta(\cdot|x_i)$
      Sample k actions $\bar{a}_i \sim \pi_{\theta'}(\cdot|x_i)$
      Compute KL divergence: $D_{\text{KL}}(x_i) \leftarrow \sum_{j=1}^k \log \frac{\pi_{\theta'}(\bar{a}_j|x_i)}{\pi_\theta(\bar{a}_j|x_i)}$
      Policy loss: $L_\pi \leftarrow \frac{1}{B} \sum Q_\phi(x_i, a_i') - e^\alpha \log \pi_\theta(a_i'|x_i) - e^\beta D_{\text{KL}}(x_i)$
      (Alternatively, compute clipped objective)
      Update actor: $\theta \leftarrow \theta + \eta_\pi \nabla_\theta L_\pi$
      Entropy α update: $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha e^\alpha (\frac{1}{B} \sum \mathcal{H}[\pi_\theta(x_i)] - \varepsilon_{\mathcal{H}})$
      KL β update: $\beta \leftarrow \beta - \eta_\beta \nabla_\beta e^\beta (\frac{1}{B} \sum D_{KL}(x_i)] - \varepsilon_{\text{KL}})$
    end
  end
  // Behavior Policy Update
  $\theta' \leftarrow \theta$
end
return *Trained policy* π_θ

RELATIVE ENTROPY PATHWISE POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Score-function based methods for policy learning, such as REINFORCE and PPO, have delivered strong results in game-playing and robotics, yet their high variance often undermines training stability. Improving a policy through state-action value functions, e.g. by differentiating Q with regard to the policy, alleviates the variance issues. However, this requires an accurate action-conditioned value function, which is notoriously hard to learn without relying on replay buffers for reusing past off-policy data. We present an on-policy algorithm that trains Q-value models purely from on-policy trajectories, unlocking the possibility of using pathwise policy updates in the context of on-policy learning. We show how to combine stochastic policies for exploration with constrained updates for stable training, and evaluate important architectural components that stabilize value function learning. The result, Relative Entropy Pathwise Policy Optimization (REPPO), is an efficient on-policy algorithm that combines the stability of pathwise policy gradients with the simplicity and minimal memory footprint of standard on-policy learning. Compared to state-of-the-art on two standard GPU-parallelized benchmarks, REPPO provides strong empirical performance at superior sample efficiency, wall-clock time, memory footprint, and hyperparameter robustness.

1 INTRODUCTION

Most modern on-policy algorithms, such as TRPO (Schulman et al., 2015) or PPO (Schulman et al., 2017), use a score-based gradient estimator to update the policy. These methods have proven useful 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), but are often plagued by training instability. Zeroth-order, score-based gradient approximation exhibits high variance (Greensmith et al., 2004), which leads to unstable learning (Ilyas et al., 2020; Rahn et al., 2023), especially in high-dimensional continuous spaces (Li et al., 2018). In addition, it requires importance sampling to allow sample reuse, which exacerbates the high variance.

An alternative, commonly used in off-policy learning, is to learn a parameterized state-action value function (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018), and use it to improve the policy, for example by using a pathwise policy gradient (Silver et al., 2014). Using a parameterized surrogate function to improve the policy often leads to faster and more stable learning learning by reducing the score-based estimators variance (Mohamed et al., 2020) and by allowing us to remove importance sampling corrections.

However, the effectiveness of these approaches is bounded by the quality of the approximate value function (Silver et al., 2014). As such, algorithms that use a state-action value function usually rely on improving value learning through off-policy training (Fujimoto et al., 2018; Haarnoja et al., 2018). Unfortunately, off-policy training requires the use of replay buffers. Storing these replay buffers can be a challenge when the collected samples cannot fit in memory. In addition, training with past data introduces various 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 our core question:

Can we train a strong surrogate value function and effectively use it for policy improvement in a fully on-policy setting without large replay buffers?

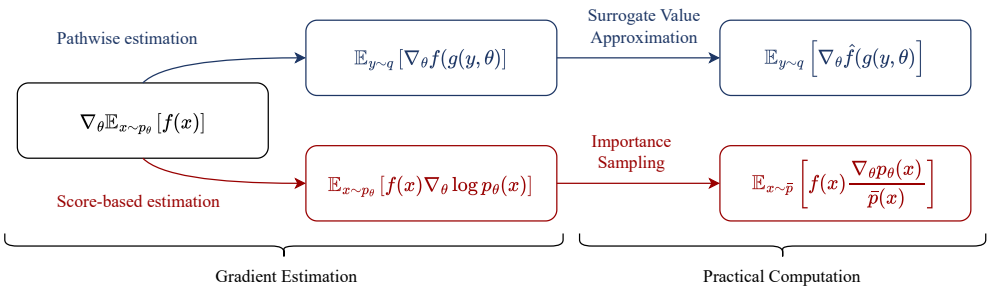


Figure 1: Overview of the strategies used by REPPO and PPO to obtain policy gradient estimators. Computing the gradient requires a mathematical transformation that allows for efficient estimation from samples, and additional steps that make the computation tractable in practice.

Building 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), we present an efficient on-policy algorithm, *Relative Entropy Pathwise Policy Optimization (REPPO)*, which uses the pathwise gradient estimator with an accurate surrogate value function learned from on-policy data. REPPO builds on the maximum entropy framework (Ziebart et al., 2008) to encourage exploration. It combines this with a KL regularization scheme, inspired by the Relative Entropy Policy Search method (Peters et al., 2010), which prevents aggressive policy updates from destabilizing the optimization.

Furthermore, we evaluate several prominent advances in neural network architecture design to stabilize learning: categorical Q-learning (Farebrother et al., 2024), normalized neural network architectures (Nauman et al., 2024a; Hussing et al., 2024), and auxiliary tasks (Jaderberg et al., 2017). These components feature in many recent variants (Schwarzer et al., 2021; 2023; Nauman et al., 2024a; Hussing et al., 2024; Gallici et al., 2024; Lee et al., 2025a;b; Nauman et al., 2025; Fujimoto et al., 2024) of common value learning algorithm such as SAC (Haarnoja et al., 2018). We find that categorical Q-learning and normalization have a strong impact on the performance, while auxiliary tasks only show small impact, but become more relevant when reducing the amount of samples.

We test our approach in a variety of locomotion and manipulation environments from the Mujoco Playground (Zakka et al., 2025) and ManiSkill (Tao et al., 2025) benchmarks, and show that REPPO is competitive with tuned on-policy baselines in terms of sample efficiency and wall-clock time, 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. In introducing REPPO, our work makes the following contributions:

1. We showcase that using a state-action value function and a pathwise policy gradient can be effective in on-policy RL, as it allows on-policy action resampling, forgoing importance corrections. However, this requires learning a highly accurate state-action value function.
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.

We provide sample implementations in both the JAX (Bradbury et al., 2018) and PyTorch (Paszke et al., 2019) frameworks. Our code is available in the supplementary material of the submission.

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. We write $\mathcal{P}_\pi(x'|x)$ for the policy-conditioned transition kernel and $\mathcal{P}_\pi^n(y|x)$ for the n-step transi-

tion kernel. 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, $J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)\right]$, 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 $Q^\pi(x, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \middle| x_0 = x, a_0 = a\right]$. We use $\mu_\pi(y|x)$ to denote the discounted stationary distribution over states y when starting in state x. When $x \sim \mu_\pi(\cdot|y)$, $y \sim \rho_0$, we will simply write $\mu_\pi(x)$ to denote the probability of a state under the discounted occupancy distribution.[1]

2.1 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

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{x\sim\mu_\pi, a\sim\pi_\theta(\cdot|x)}[Q^{\pi_\theta}(x, a)\nabla_\theta \log \pi_\theta(a|x)]. \tag{1}$$

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.

An alternative approach, leveraged in off-policy learning, is the *deterministic policy gradient theorem* (DPG) (Silver et al., 2014). The estimator for the DPG relies on access to a differentiable state-action value function and a deterministic differentiable policy $\pi_\theta^{\text{det}}(x)$. While access to the true value function is an unrealistic assumption, we can use a trained surrogate model, $\hat{Q}$, to obtain a biased estimate of the gradient

$$\nabla_\theta J(\pi_\theta) \approx \mathbb{E}_{x\sim\mu_\pi}[\nabla_a \hat{Q}^{\pi_\theta^{\text{det}}}(x, a)|_{a=\pi_\theta^{\text{det}}(x)} \nabla_\theta \pi_\theta^{\text{det}}(x)]. \tag{2}$$

Finally, the DPG can be expanded to reparameterizeable stochastic policies[2]. We term this the *pathwise policy gradient*, following Mohamed et al. (2020), but the formulation has been used prominently in prior work such as SAC (Haarnoja et al., 2018), just without a proper name. The gradient estimator can be obtained from the following expectation

$$\nabla_\theta J(\pi_\theta) \approx \mathbb{E}_{x\sim\mu_\pi, \epsilon\sim p(\epsilon)}[\nabla_a \hat{Q}^{\pi_\theta^{\text{rep}}}(x, a)|_{a=\pi_\theta^{\text{rep}}(x,\epsilon)} \nabla_\theta \pi_\theta^{\text{rep}}(x, \epsilon)], \tag{3}$$

where $\pi_\theta^{\text{rep}}(x, \epsilon)$ is a reparameterization of $\pi_\theta(a|x)$. To avoid notational we will write $\pi_\theta(a|x)$ from now on to always mean the appropriate reparameterization.

2.2 Understanding sources of harmful variance in gradient estimation

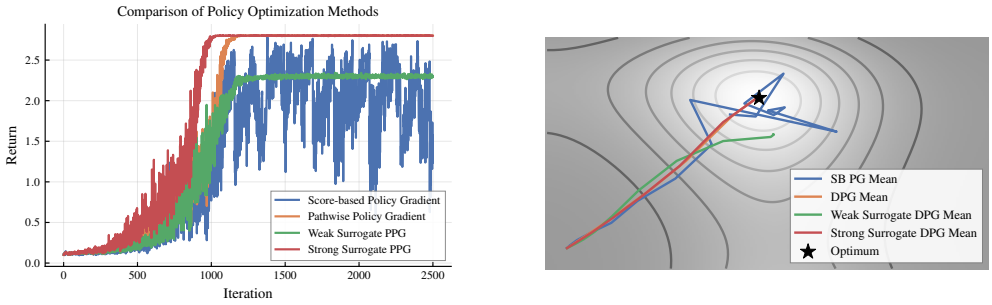
To build additional intuition on the differences between different policy gradient estimators, we conduct an illustrative experiment. Implementation details can be found in Appendix D.

On a simple objective $g(x)$ we initialize four Gaussians and update their parameters to maximize $J(\mu, \Sigma) = \mathbb{E}_{x\sim\mathcal{N}(\cdot|\mu,\Sigma)}[g(x)]$ with four different methods: a score-based policy gradient (using Equation 1), 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 3). We visualize the returns and the path of the mean estimates in Figure 2a. In addition, we zoom in on the gradient paths of the score-based estimator. We visualize 100 different eight step paths from the middle of the trajectory. Here, in addition to the vanilla score-based estimator, we also show an importance sampling and a clipped importance sampling estimator. These paths are visualized in Figure 2b.

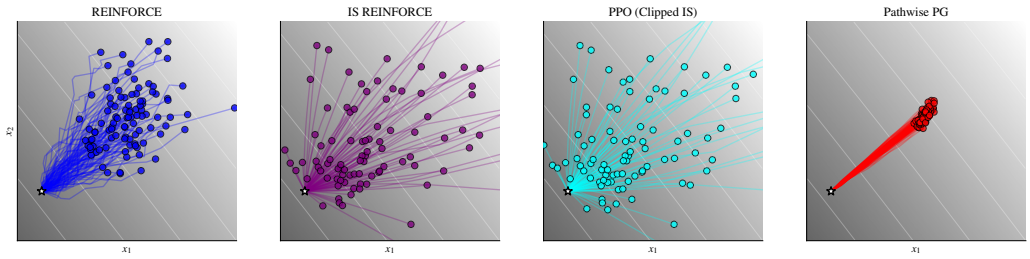
The experiments shows that score-based gradient estimators have high variance, and can lead to unstable policies which fail to optimize the target. In addition, while importance sampling increases the sample efficiency of the algorithm, it greatly exacerbates these variance issues. We find that clipping the ratio estimate, as proposed by Schulman et al. (2017), prevents catastrophic instability,

[1] A well-known issue of many policy gradient works is that in practice, they, perhaps erroneously, use the undiscounted empirical state occupancy for optimization (Nota & Thomas, 2020). REPPO similarly uses empirical samples without accounting for the discount factor in the objective.

[2] We discuss an extension to non-reparametrizeable, discrete policies in Appendix C.



(a) Achieved returns (left) and path of four policies trained with different gradient estimation methods. We compare a score-function 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.



(b) Gradient path over eight steps in the middle of the trajectory, visualized per algorithm for 8 steps. For Reinforce and PPG, new samples are drawn at every step. For the importance sampling based algorithms, one set of samples is sampled at the beginning and subsequent steps are conducted using importance sampling.

Figure 2: Visualization of gradient paths on a 2D example function.

but does not reduce the variance substantially. On the other hand, using a pathwise gradients is remarkably stable and exhibits small variance. However, it either requires access to the gradients of the objective function, or a strong surrogate model.

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

We now present our algorithm for using 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, as 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. Combining a set of recent advances in both reinforcement learning as well as neural network value function fitting, can satisfy these requirements. We first introduce the core RL algorithm, and then elaborate on the architectural design of the method.

At its core, REPPO proceeds similar to other on-policy actor-critic algorithms through three distinct phases: data gathering, value target estimation, and value and policy learning (see Algorithm 1). To obtain diverse data, REPPO uses a maximum-entropy formulation, adapted to multi-step TD-λ (Subsection 3.1), to encourage exploration. 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 and policy constraints (Subsection 3.2).

3.1 Value function learning

Off-policy PPG methods like TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018) mostly use single step Q learning, i.e. they use only immediate rewards for value function updates. This

is paired with large replay buffers to stabilize learning. While on-policy algorithms cannot use past policy data, they can instead use low bias multi-step TD targets for stabilization (Fedus et al., 2020). Therefore, multi-step TD-λ targets form the basis for our value learning objective. Note that REPPO is more closely related to SARSA than to Q-learning (Sutton & Barto, 2018), due to being on-policy.

In addition to multi-step returns, 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 not only maximizes rewards but also penalizes the loss of entropy in the policy distribution. The maximum-entropy policy objective (Levine, 2018) can be defined as

$$J_{\mathrm{ME}}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) + \alpha \mathcal{H}[\pi_\theta(x_t)]\right], \tag{4}$$

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

$$G^{(n)}(x_t, a_t) = \sum_{k=t}^{n-1} \gamma^k (r(x_k, a_k) - \alpha \log \pi(a_k|x_k)) + \gamma^n Q(x_n, a_n) \tag{5}$$

$$G^{\lambda}(x, a) = \frac{1}{\sum_{n=0}^{N} \lambda^n} \sum_{n=0}^{N} \lambda^n G^{(n)}(x, a), \tag{6}$$

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). Crucially, the targets are computed on-policy after a new data batch is gathered, and the Q targets are not recomputed before gathering new data. Our Q learning loss is

$$\mathcal{L}_Q^{\mathrm{REPPO}}\left(\phi | \{x_i, a_i\}_{i=1}^B\right) = \frac{1}{B} \sum_{i=1}^{B} \mathrm{HL}\left[Q_\phi(x_i, a_i), G^\lambda(x_i, a_i)\right] + \mathcal{L}_{\mathrm{aux}}(f_\phi(x_i, a_i), x_i'), \tag{7}$$

where x_i' refers to the next state sample starting from x_i, and HL is the HL-Gauss loss (see Subsection 3.3 and Subsection D.2), and $\mathcal{L}_{\mathrm{aux}}$ is presented in Subsection 3.3 and Subsection D.3.

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), so this simplification increases the robustness of our method. We also do not need a target value function copy, since we do not recompute the target at each step and it therefore remains on-policy.

3.2 Policy Learning

A core problem with value-based on-policy optimization is controlling the size of the policy update, as the value estimate is only accurate on the data covered by the prior policy. A large policy update can therefore destabilize learning (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 the behavioral (Peters et al., 2010; Schulman et al., 2015). To control the deviation, we use the Kullback-Leibler (KL) divergence, also called the relative entropy (Peters et al., 2010), as it can be justified theoretically through information geometry (Kakade, 2001; Peters & Schaal, 2008; Pajarinen et al., 2019), and is easy to approximate using samples.

Some works in the literature (Neumann, 2011; Sokota et al., 2022) claim that the reverse mode might be preferable for policy constraints, as it is mode-seeking, and the forward mode is mode-averaging. However, this intuition does not cleanly translate to our setting. As our policies are unimodal tanh-squashed Gaussian, the main impact of the KL direction is that the reverse-mode KL is entropy reducing. As we explicitly aim to increase the policy's entropy using the maximum entropy formulation, using forward-mode KL makes the optimization more stable.

Policy Optimization Objective Our policy updates derive from a constrained optimization problem which includes both entropy and the KL constraint, and where θ' is the behavior policy, and $\varepsilon_{\mathrm{KL}}$ and $\varepsilon_{\mathcal{H}}$ are the respective KL and entropy constraints

$$\max_{\theta} \quad \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|x)}\left[Q(x,a)\right]\right] \tag{8}$$

$$\text{subject to} \quad \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[D_{\mathrm{KL}}\left(\pi_{\theta'}(\cdot|x)\,\|\,\pi_\theta(\cdot|x)\right)\right] \leq \varepsilon_{\mathrm{KL}} \tag{9}$$

$$\mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[\mathcal{H}[\pi_\theta(\cdot|x)]\right] \geq \varepsilon_{\mathcal{H}}. \tag{10}$$

A similar combination of maximum entropy and KL divergence bound has been explored in various forms (Abdolmaleki et al., 2015; Pajarinen et al., 2019; Akrour et al., 2019). However, while previous approaches use complex solutions to this problem, such as approximate mirror descent, line search, or heuristic clipping, we take a simpler approach. We relax the problem, which introduces two hyperparameters, α for the entropy, and β for the KL. Inspired by Haarnoja et al. (2019), REPPO automatically adapts these constraints when the policy violates them.

Policy Updates and Multiplier Tuning In the constrained objective, we introduce two hyperparameters, $\varepsilon_{\mathcal{H}}$ and $\varepsilon_{\mathrm{KL}}$, which bound the entropy and KL divergence. The goal of the 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

$$\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha e^\alpha \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[\left(\mathcal{H}[\pi_\theta(\cdot|x)] - \varepsilon_{\mathcal{H}}\right)\right] \tag{11}$$

$$\beta \leftarrow \beta - \eta_\beta \nabla_\beta e^\beta \mathbb{E}_{x\sim\rho_{\pi_{\theta'}}}\left[\left(\mathrm{D}_{\mathrm{KL}}(\pi_{\theta'}(\cdot|x)\|\pi_\theta(\cdot|x)) - \varepsilon_{\mathrm{KL}}\right)\right]. \tag{12}$$

Finally, to ensure our KL constraint is (approximately) maintained, we clip the actor loss based on whether the constrained is currently violated. The full policy objective for REPPO is now

$$\mathcal{L}_\pi^{\mathrm{REPPO}}(\theta|x_i) = \begin{cases} -Q(x_i,a) + e^\alpha \log \pi_\theta(a|x_i), & \text{if } \frac{1}{k}\sum_{j=1}^k \log \frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} < \varepsilon_{\mathrm{KL}} \\ e^\beta \frac{1}{k}\sum_{j=1}^k \log \frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)}, & \text{otherwise} \end{cases} \tag{13}$$

where a is sampled from $\pi_\theta(\cdot|x_i)$ and a_j from the past behavior policy $\pi_{\theta'}(\cdot|x_i)$, and k denotes how many samples are used to approximate the KL. As with the critic, the optimized loss is a mean over a minibatch from the rollout data. Note that contrary to other on-policy algorithms like PPO and TRPO, we are not forced to use actions sampled from the behavior policy in the policy gradient estimator, which removes the need for importance sampling correction. We will show that this greatly improves the performance of REPPO in Subsection 4.1.

Jointly tuning the entropy and KL multipliers is a crucial component of REPPO. As the policy entropy and KL are tied, letting the entropy of the behavior policy collapse results in a scenario where 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, 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.

3.3 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.

Cross-entropy loss for regression The first choice is to replace 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), which can 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. Palenicek et al. (2025) further investigate and reinforce this claim, showing that stable gradients arise from cross-entropy based losses. We present the mathematical form of the loss formulation in Subsection D.2. We find that a categorical

loss is a crucial addition, as our ablation experiments show (Subsection E.1), but alternatives like C51 could easily work as well.

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 shown the importance of layer normalization (Ba et al., 2016) for stable critic 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 off-policy bootstrapped methods; yet, we still see performance benefits in most environments from normalization.

Auxiliary tasks Auxiliary tasks (Jaderberg et al., 2017) can stabilize features in environments with sparse rewards, where the lack of a reward signal can prevent learning meaningful representations via the Q learning objective (Voelcker et al., 2024a). For REPPO, auxiliary tasks are especially impactful when we decrease the number of samples used in each update batch (see Subsection E.1). We provide a discussion of this auxiliary task setup, including the loss function, in Subsection D.3.

4 Experimental Evaluation

We begin by evaluating whether pathwise estimators improve upon score-based estimation in on-policy RL settings. We then compare our approach to baselines, evaluating final performance, sample and wall-clock efficiency, and stability of policy improvement. Our results demonstrate strong performance of REPPO on all axes. Additional details on architectures, hyperparameters, and ablations are provided in Subsection D.4 and Appendix E. A discrete variant of REPPO, along with its architectural changes and experimental results, is presented in Appendix C.

Environments We evaluate REPPO on two major GPU-parallelized benchmark suites: 23 tasks from the mujoco_playground DMC suite (Zakka et al., 2025) and 8 ManiSkill environments (Tao et al., 2025), covering locomotion and manipulation, respectively. These tasks span high-dimensional control, sparse rewards, and chaotic dynamics.

4.1 Score-based and Pathwise Comparison

REPPO offers an alternative to score-based policy gradient estimation in on-policy RL. However, we also introduce several enhancements, including automated tuning of entropy and KL coefficients, to improve value and policy learning. To assess the benefits of learned values and pathwise gradient estimation over score-based methods, we conduct two experiments. First, we replace the pathwise term $-Q(x, a)$ in Equation 13 with the score function $\log \pi(a|x)[Q(x, a)]_{\text{sg}}$, denoted as *REPPO (score-based, Q)*. Second, we replace the gradient estimator with the GAE-based clipped objective from PPO, denoted as *REPPO (score-based, GAE)*. Aggregate results are presented in Figure 3.

Using the approximate Q function in the policy gradient objective provides a strong improvement over PPO or REPPO with a clipped objective. Q score-based REPPO outperforms PPO, strongly showcasing the benefits of value function learning and removing importance sampling. This also shows that the REPPO framework can be used with policy classes that are not amenable to reparameterization, such as diffusion policies (Chi et al., 2024; Celik et al., 2025; Ma et al., 2025), by

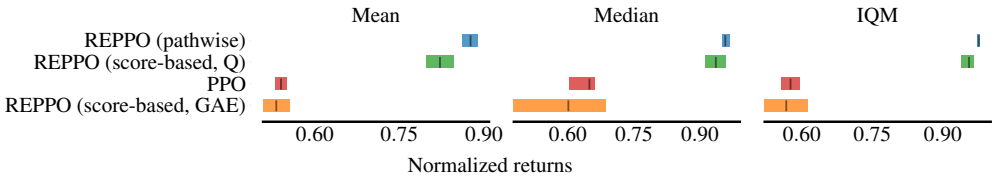
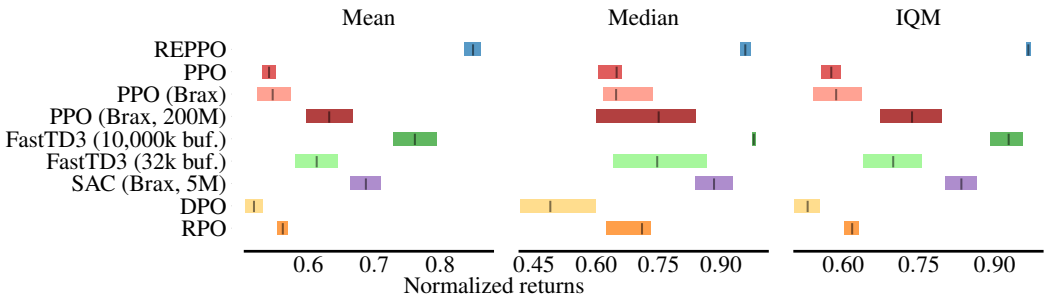
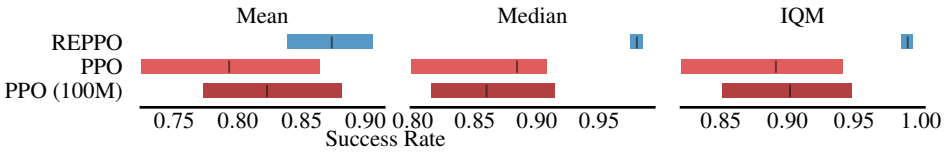


Figure 3: Aggregate performance metrics on the mujoco_playground benchmark. We compare REPPO with two ablations: one using the score-based gradient estimator with the learned Q function, and another using an on-policy GAE estimate with importance sampling and clipping. For additional context, we also report PPO results.



(a) Aggregate performance metrics on the mujoco_playground DeepMind Control Suite benchmark. We compare both REPPO and our 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), as well as FastTD3 (Seo et al., 2025), RPO (Rahman & Xue, 2023), and DPO (Lu et al., 2022).



(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 than 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 4: Aggregate performance comparison on (a) mujoco_playground DMC and (b) ManiSkill3.

using a score-based estimator together with the learned Q function. Interestingly, combining the PPO objective with REPPO leads to slightly worse results than vanilla PPO. We find that the high variance complicates the automatic parameter tuning scheme.

4.2 Benchmark comparison

We compare REPPO against the PPO and SAC results reported by Zakka et al. (2025) and Tao et al. (2025). We report PPO baselines at 50M environment steps, and at the larger training horizon used in the original papers (Zakka et al., 2025). Results taken from Zakka et al. (2025) are denoted as "PPO/SAC (Brax)". To ensure that PPO is not undertuned for the 50m step regime we re-tuned the hyperparameters of the implementation provided by Lu et al. (2022). SAC results are reported at 5m steps as this amounts to similar total runtime as the 200m PPO results (compare results in Zakka et al. (2025). Naively running SAC at a larger sample budget and wall-clock efficiency can lead to instability, as Seo et al. (2025) demonstrates. Furthermore, we include FastTD3 (Seo et al., 2025) on DMC locomotion tasks, trained under two memory budgets: the default replay buffer (10,485,760 transitions) and a constrained buffer similar in size to on-policy methods (32,768 transitions) to control for the the memory and performance trade-off. Finally, we compare against Robust Policy Optimization (RPO) (Rahman & Xue, 2023) and Discovered Policy Optimization (DPO) (Lu et al., 2022). However, even with some hyperparameter tuning, we were unable to achieve a strong performance improvement beyond the PPO baseline with these approaches.

For REPPO, we report results aggregated over 20 seeds across all tasks. We run 20 seeds for PPO and 5 for FastTD3[3], reporting aggregate scores with 95% bootstrapped confidence intervals (Agarwal et al., 2021). To enable aggregation across tasks, returns on mujoco_playground are normalized by the maximum achieved by any algorithm, while for ManiSkill we report raw success rates, which are naturally comparable across tasks.

[3]We use fewer seeds for FastTD3 as we are unable to replicate the speed claimed in the paper. This is due pytorch specific issues discussed in Appendix B, and because we use smaller GPUs for our experiments.

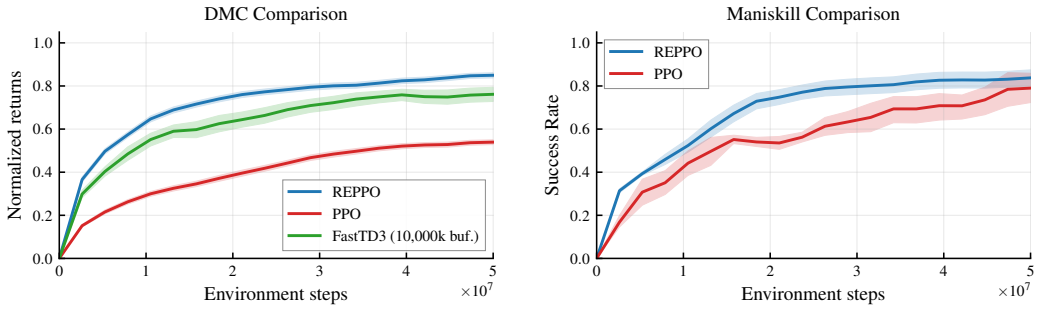


Figure 5: Aggregate sample efficiency curves for the benchmark environments. Settings are identical to those in Figure 4. REPPO achieves higher performance at a faster rate in both benchmarks.

Final Performance and Sample Efficiency We first investigate the performance of policies trained using REPPO. We report aggregate performance at the end of training on both benchmarks in Figure 4. For both benchmarks, we also provide the corresponding training curves in Figure 5.

The aggregate results shown in Figure 4 and Figure 5 indicate that our proposed method achieves statistically significant performance improvements over PPO, 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 (Chan et al., 2020a; Agarwal et al., 2021).

We find that PPO struggles on high-dimensional tasks such as HumanoidRun, even with large batch sizes aimed at reducing policy gradient variance. Moreover, despite its approximate trust-region updates, PPO suffers from performance drops and unstable training. This erratic behavior closely mirrors the score-based policy gradient instability shown in Figure 2a. In contrast, REPPO exhibits more stable improvements and lower variance across seeds.

Wall-clock Time Wall-clock time is an important metric, as it reflects the practical utility of an algorithm: faster training enables more efficient hyperparameter search and experimentation. However, measuring wall-clock time is nuanced, as results heavily depend on implementation details and are difficult to reproduce. We discuss these challenges across different frameworks in Appendix B. In Figure 6, we compare the wall-clock performance of our approach against PPO and SAC in JAX. Other baselines lack JIT-compilable implementations, making direct comparisons less meaningful.

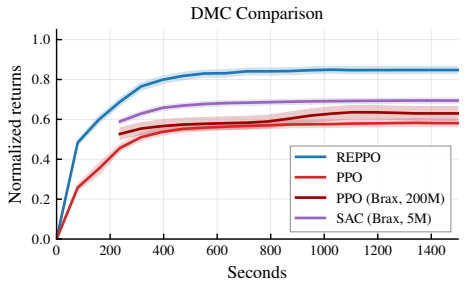


Figure 6: Wall-clock time comparison of REPPO against PPO and SAC implementations in JAX. REPPO matches other algorithms' speed but achieves higher return.

The computational cost per update is higher for REPPO than for PPO due to larger default networks and gradient propagation through the critic–actor chain. Nevertheless, both algorithms converge on most tasks in roughly 600–800 seconds, with REPPO achieving about 33% higher normalized returns. This shows that the sample efficiency of pathwise gradients can offset their higher per-update cost, yielding improved wall-clock efficiency compared to score-based PPO. In addition, we find that jax-based SAC, which is tuned to trade sample for computational efficiency, slightly outperforms PPO, but does not match REPPO in performance. We note that other, modern SAC implementations (Nauman et al., 2024b; Lee et al., 2025a;b), are able to achieve better performance, but at the cost of computational efficiency.

Reliable Policy Success We further investigate the stability of policy improvements using score-based and pathwise policy gradients. Our guiding principle is that such updates should not cause large drops in performance. To capture this, we adopt the "reliable success" metric, as proposed in Chan et al. (2020b). We define an algorithm as *reliably performant* if, once its performance

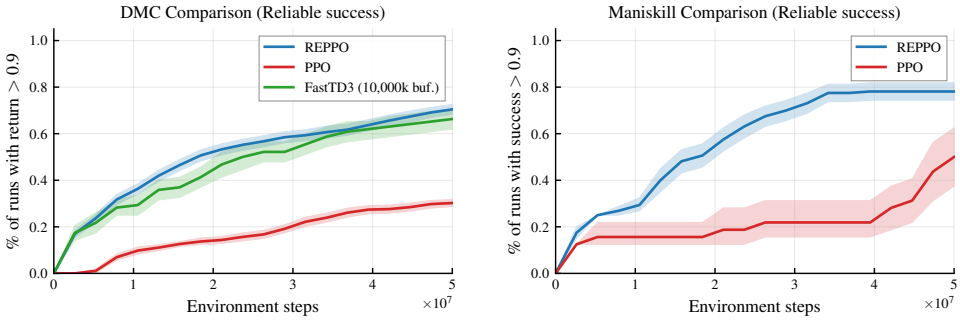


Figure 7: Fraction of runs that achieve reliable performance as measured by our metric for policy stability and reliability. REPPO's immediately starts achieving high performance in some runs and the number gradually increases indicating stable learning. PPO struggles to achieve high performance initially and to maintain high performance throughout training.

exceeds a fixed threshold τ, it never drops below this threshold thereafter. At each timestep, we track the number of runs that satisfy this criterion. This metric reflects the practical requirement that a deployed algorithm should not suddenly degrade simply due to continued training. We report the percentage of reliably successful runs for both REPPO and PPO in Figure 7.

On both DMC and ManiSkill benchmarks, REPPO achieves reliable performance improvements quickly, with success rates and returns steadily increasing. By the end of training, about four out of five runs have reached the threshold of $\tau = 0.9$ without dropping below it, whereas PPO achieves roughly 40 percentage points fewer reliably performant runs. We also find notable differences in sample efficiency: PPO requires 5–10 million interactions before most envs become reliably performant. Overall, these results show that, despite relying on a biased surrogate value model, pathwise policy gradients enable stable long-term improvement.

5 Conclusion and Avenues for Future Work

In this paper we present REPPO, a highly performant yet efficient on-policy algorithm that leverages trained state-action value functions and pathwise 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.

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. We also observe that removing the importance sampling step in PPO has a crucial impact on performance, which suggests further research on the trade-off between efficiency and stability in on-policy gradient estimation is needed. 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.

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 *Proceedings of the 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 *Advances in Neural Information Processing Systems*, 2022.

Riad Akrour, Joni Pajarinen, Jan Peters, and Gerhard Neumann. Projections for approximate policy iteration algorithms. In *Proceedings of the 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.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. DIME: Diffusion-based maximum entropy reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2025.

Stephanie Chan, Sam Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *Proceedings of the International Conference on Learning Representations*, 2020a.

Stephanie CY Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *Proceedings of the International Conference on Learning Representations*, 2020b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

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.

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 *Proceedings of the 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 *Proceedings of the International Conference on Learning Representations*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the International Conference on Machine Learning*, 2023.

Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In *Proceedings of the 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 *Proceedings of the 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. *arXiv preprint arXiv:1812.05905*, 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 *Proceedings of the International Conference on Learning Representations*, 2024.

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 *Proceedings of the International Conference on Learning Representations*, 2020.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *Proceedings of the 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.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*, 2017.

Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip Thomas. Evaluating the performance of reinforcement learning algorithms. In *Proceedings of the International Conference on Machine Learning*, 2020.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 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 *Proceedings of the 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 *Proceedings of the 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 International Joint Conference on Artificial Intelligence*, 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*, 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. In *Proceedings of the Conference on Lifelong Learning Agents*, 2024.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Proceedings of the International Conference on Machine Learning*, 2025.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*, 2017.

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.

Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. No representation, no trust: Connecting representation, collapse, and trust issues in PPO. In *Advances in 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 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.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=zhOUfuOIzA.

Gerhard Neumann. Variational inference for policy search in changing situations. In *Proceedings of the International Conference on International Conference on Machine Learning*, 2011.

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.

Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

Fabian Otto, Philipp Becker, Vien Anh Ngo, Hanna Carolin Maria Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2021.

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, 2022.

Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible natural gradient policy search. *Machine Learning*, 108(8), 2019.

Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. *arXiv preprint arXiv:2509.25174*, 2025.

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*, 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 AAAI Conference on Artificial Intelligence*, 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.

Md Masudur Rahman and Yexiang Xue. Robust policy optimization in deep reinforcement learning, 2023.

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, 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. *arXiv preprint arXiv:2505.22642*, 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.

Samuel Sokota, Ryan D'Orazio, J Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, Noam Brown, and Christian Kroer. A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Sanghyun Son, Laura Yu Zheng, Ryan Sullivan, Yi-Ling Qiao, and Ming Lin. Gradient informed proximal policy optimization. In *Advances in Neural Information Processing Systems*, 2023.

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 *Proceedings of the International conference on Learning Representations*, 2019.

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*, 2022.

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, 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 *Proceedings of the 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.

Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Proceedings of the 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 *Proceedings of the International Conference on Learning Representations*, 2022.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving q-value divergence in offline-RL. In *Advances in 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 National Conference on Artificial Intelligence*, 2008.

A Extended 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).

Incorporating ground-truth gradient signal to stabilize training has also 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., 2022).

Trust regions and constrained policy optimization Other approaches have used similar KL and trust region constraint as REPPO. 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). Akrour et al. (2019) propose to project the policy onto the trust-region to sidestep the difficulty associated with clipping. We find that our approach is simpler to implement and more general, as we do not assume direct projection is possible.

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.

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. We also evaluate our approach on discrete action spaces (see Appendix C). While PQN performs slightly better, likely owing to tuned exploration techniques, we show that our method works robustly across both discrete *and* continuous action spaces.

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.

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 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. This is the reason why we do not include FastTD3 in Figure 6, as only a PyTorch implementation of the algorithm exists. FastTD3 and REPPO use similar algorithms and hyperparameters, therefore, barring complexities like those discussed below, we expect them to perform at similar speeds.

More importantly, torch's compilation libraries are built to accelerate standard supervised and generative workflows, but do not support RL primitives equally well. As the CPU needs to load kernels during training which the GPU then executes, the CPU plays a much larger role in the speed measurements of the torch-based variant of REPPO. Especially the tanh-squashed log probability computation and the frequent resampling from the action space cannot be offloaded into an efficient

kernel without providing one manually, which we have not done. This is likely due to the fact that torch keeps its random seed on the CPU. This is not a concern for jax, due to the fact that all kernels are statically compiled when the program is first executed, and random seeds are handled explicitly as part of the program state. Therefore, 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.

C DISCRETE REPPO (D-REPPO)

One of the major advantages of PPO in the zoo of RL algorithms is the fact that it can be used in both continuous and discrete action settings. However, as we build on the DDPG/TD3/SAC line of work, the exposition of our algorithm has focused on the continuous setting alone.

Nonetheless, it is easy to adapt our approach to the discrete action setting as well. Following the proposal of Christodoulou (2019), we can circumvent the chained critic-actor gradient and compute the value of the current policy, the entropy, and the KL bound in closed form

$$\mathcal{L}^{\mathrm{D-REPPO}}_{\pi,\leq\mathrm{KL}}(\theta|B) = -\frac{1}{|B|}\sum_{i=1}^{|B|}\sum_{j=1}^{|A|}\pi_\theta(a_j|x_i)\left(Q(x_i,a_j) + e^{\alpha}\log\pi_\theta(a_j|x_i)\right) \tag{14}$$

$$\mathcal{L}^{\mathrm{D-REPPO}}_{\pi,>\mathrm{KL}}(\theta|B) = -\frac{1}{|B|}\sum_{i=1}^{|B|}e^{\beta}\sum_{j=1}^{|A|}\pi_{\theta'}(a_j|x_i)\log\frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} \tag{15}$$

$$\mathcal{L}^{\mathrm{D-REPPO}}_{\pi}(\theta|B) = \begin{cases}\mathcal{L}^{\mathrm{D-REPPO}}_{\pi,\leq\mathrm{KL}}(\theta|B), & \text{if } \sum_{j=1}^{k}\log\frac{\pi_{\theta'}(a_j|x_i)}{\pi_\theta(a_j|x_i)} < \varepsilon_{\mathrm{KL}} \\ \mathcal{L}^{\mathrm{D-REPPO}}_{\pi,>\mathrm{KL}}(\theta|B), & \text{otherwise.}\end{cases} \tag{16}$$

This variant of our algorithm still directly differentiates the full Q function objective, so can still be seen as a pathwise implementation. But computing the expectation in closed form circumvents the necessity to use a biased estimator for discrete sampling, such as the Gumbel-Softmax trick (Maddison et al., 2017; Jang et al., 2017; Fujimoto et al., 2024).

To investigate the benefits of our approach in the discrete action setting, we compare it against PQN (Gallici et al., 2024) and PPO. The main benefit of our approach over PQN is that it is a) a general algorithm that unifies both discrete and continuous action spaces, due to the underlying actor critic architecture, and b) that the principled entropy and KL objectives stabilize updates and encourages continuing exploration without an epsilon greedy exploration strategy.

We find that our algorithm is able to perform roughly on-par with PQN in the Atari-10 suite of games (cf. Table 1 and Figure 8) with only minor changes to the architecture to adapt to the Atari games benchmark. Notably, suitable settings for the KL and entropy target remain consistent even for the discrete action setting. We only find that the value of $\lambda = 0.65$ that is also recommended by Gallici et al. (2024) is superior to our default value of 0.95, likely due to the higher variance of the return in the atari games. While the high variance across Atari games makes drawing a clear conclusion difficult, we find that PQN seems to achieve slightly better performance. We find that this is most likely due to the fact that the algorithm adds explicit exploration noise, while we rely on the entropy and conservative KL terms to pace policy improvement.

Table 1: Aggregated Human-Normalized Atari-10 scores with 95% confidence intervals.

Algorithm	**Mean [CI]**	**Median [CI]**	**IQM [CI]**
REPPO	2.98 [2.64, 3.33]	1.68 [1.48, 1.82]	1.64 [1.54, 1.74]
PQN	3.35 [3.00, 3.76]	1.58 [1.48, 1.71]	1.64 [1.58, 1.71]

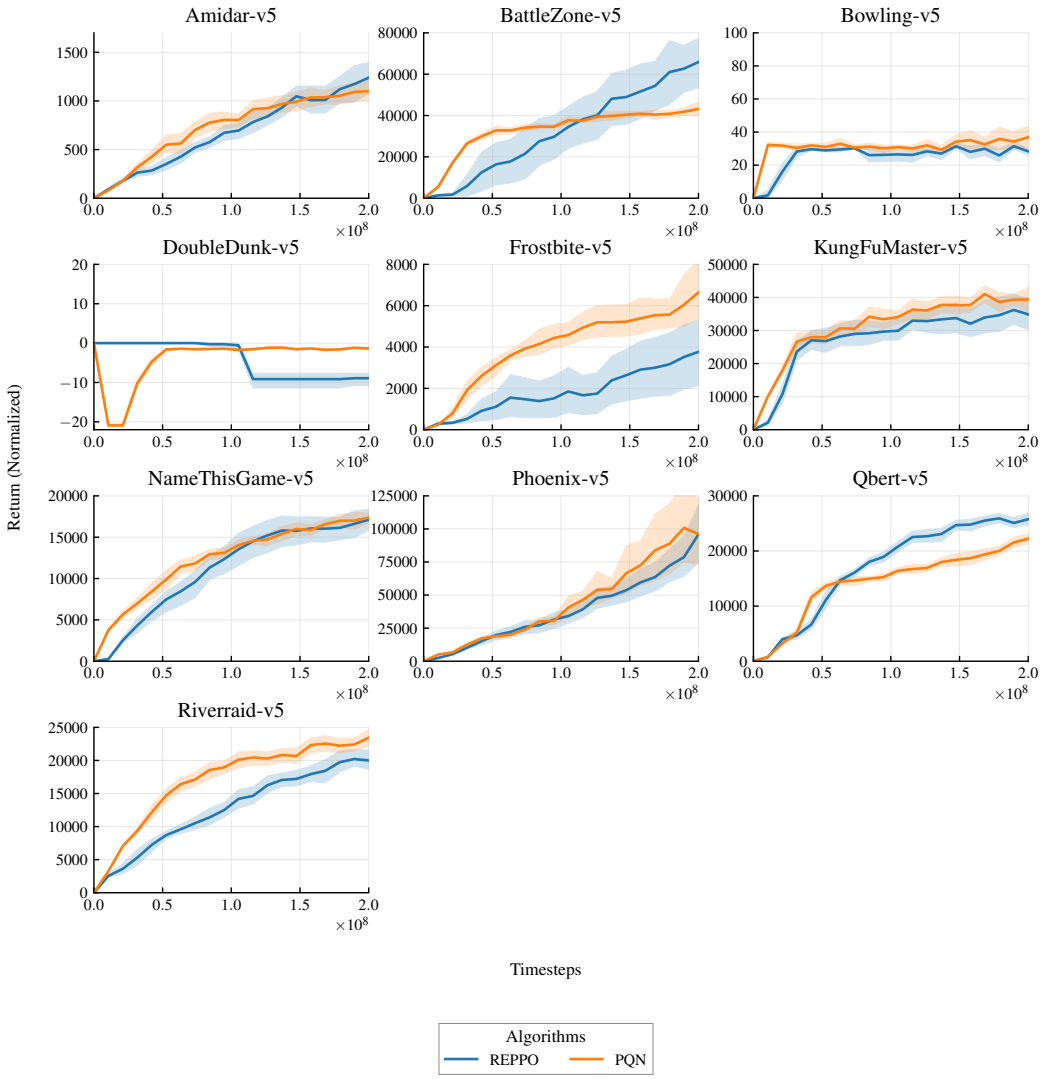


Figure 8: Per-environment results on the Atari-10 suite

D Implementation details and hyperparameters

In the following, we present implementation details on experiments, as well as a hyperparameter overview.

D.1 Toy example

To obtain the gradient descent comparison in Subsection 2.2 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

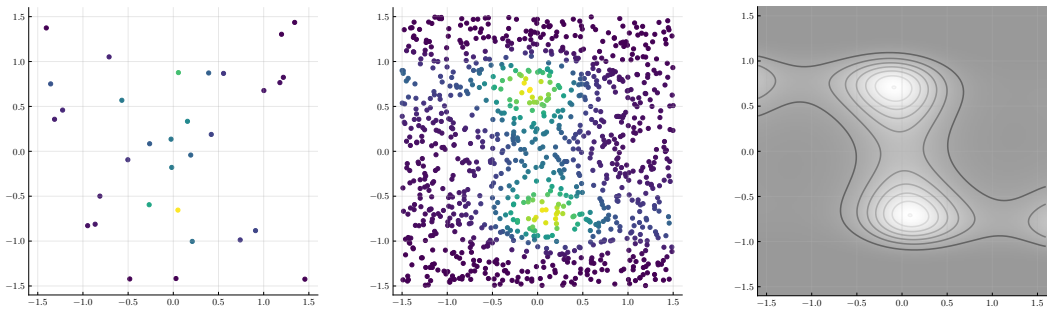


Figure 9: Samples used to train the surrogate function. On the left, we visualize the 32 sample dataset to train the weak surrogate function, in the middle the 1024 datapoints to train the strong, and on the right the full objective function.

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.2 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

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

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 $\max$.

D.3 Auxiliary Task Setup

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

$$\mathcal{L}_{\mathrm{aux}}(x_t, a_t, x_{t+1}, a_{t+1}) = \left| f_p(\phi(x_t, a_t)) - \phi(x_{t+1}, a_{t+1}) \right|^2 . \tag{17}$$

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

Environment	
total time steps	$50,000,000$
n envs	1024
n steps	128
$\mathrm{KL}_{\mathrm{tar}}$	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
$\varepsilon_{\mathrm{KL}}$	0.1
α start	0.01
$\varepsilon_{\mathcal{H}}$	$0.5 \times \dim \mathcal{A}$
aux loss mult	1.0

Table 2: Default REPPO hyperparameters

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.

D.4 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 2. 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). This makes the hyperparameters, together with the algorithm description, and the source code, a *complete algorithm specification* in the sense of Jordan et al. (2020), as we only vary hyperparameters across environments following simple equations on clear, domain sepcific hyperparameters such as the size of the action space and the length of the experiment.

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.

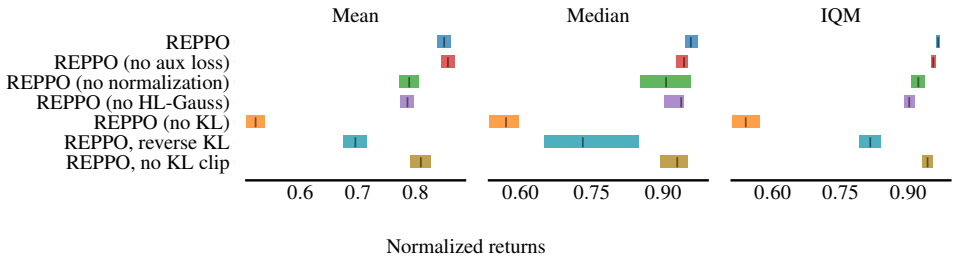
For more exact details on the architecture we refer to interested readers to the codebase.

E Additional Results

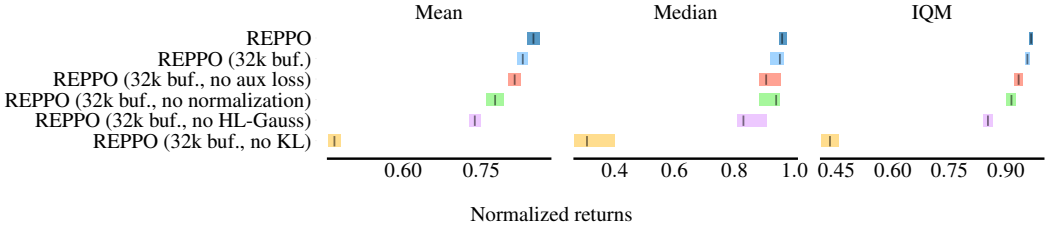
In the following, we provide additional results and further clarification on existing experiments in Section 4.

E.1 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



(a) Large dataset size ablation (128×1024).



(b) Small dataset size ablation (32×1024).

Figure 10: Ablation on components and data size on the DMC benchmark. 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.

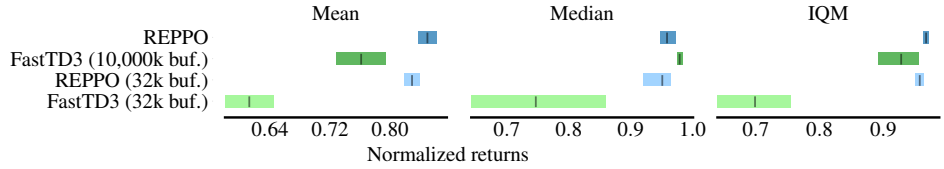


Figure 11: Comparison of aggregate performance between REPPO and FastTD3. REPPO is competitive with the large buffer FastTD3 version and outperforms FastTD3 when memory is limited.

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 $32,768$.

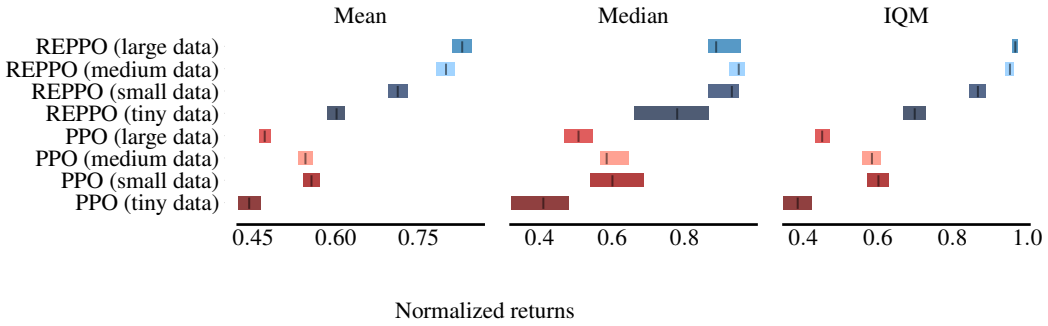
As shown in Figure 10, our results indicate that both the KL regularization of the policy updates and the categorical Q-learning via HL-Gauss are necessary to achieve strong performance independent of the size of the on-policy data used to update our model. We find that the KL divergence is the only 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 a more clearly negative 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.

E.2 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

	Num envs	Num steps	Num minibatches	Epochs	Updates per batch
Large data	1024	128	64	8	512
Medium data	1024	32	16	8	128
Small data	1024	8	4	8	32
Tiny data	256	8	1	8	8

(a) Dataset configurations for the data scaling experiment.



(b) Aggregated performance of REPPO and PPO under different batch dataset sizes. The mean performance of REPPO drops monotonically with decreasing batch size, while PPO shows its highest performance with a medium and small dataset size.

Figure 12: Experiment to compare the impact of batch datset size on different on-policy algorithms.

a large amount of data. For this, we compare against the recent FastTD3 (Seo et al., 2025) which also uses GPU-parallelized environments but operates off-policy. 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 11.

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.

E.3 Data scaling

To further understand what enables REPPO to perform well, we take a detailed look at the interplay between batch size and gradient steps. In our default configuration, REPPO uses very long rollouts and a high number of parallel environments, as well as a large number of policy and value function update steps. PPO on the other hand works best at smaller dataset sizes. We therefore set up REPPO and PPO training runs across 4 datasets, varying the rollout length. To keep the total number of gradient steps and the minibatch size the same, we reduced the number of minibatches proportionally to the batch size. The settings are summarized in Figure 12a. Note that in the large settings, the data becomes more off-policy. Both PPO and REPPO have explicit ways to deal with this, clipping and the KL minimization term respectively, but the clipping term in PPO is only a heuristic to prevent large importance sampling ratios.

Comparing the performance of both approaches (see Figure 12b), we observe a clear pattern. The mean performance of REPPO drops steeply with decreasing dataset size. PPO on the other hand does best in the medium and small dataset regimes. This highlights the different mechanisms on which both algorithms operate. Larger datasets allow the trained Q function to generalize better, similar to the insight presented in Figure 2a. On the other hand, for PPO the dataset size needs to be large enough to allow for stable gradient estimation, but not so large that too many gradient update steps are necessary. This is because clipping can prevent further learning, and many update steps can exacerbate varaince issues with importance sampling.

Note that at some point, REPPO will likely also stop improving with larger datasets and more gradient update steps. We see that the performance differences between the medium and the large dataset at not as strong as with smaller datasets. REPPO cannot continue to learn on fixed data forever, by design, as the KL divergence between two consecutive policies is constrained. However, we can hypothesize based on the empirical evidence that REPPO is able to scale more gracefully with large amounts of data.

E.4 Per Environment Sample Efficiency Curves

Finally, we provide sample efficiency curves per environment in Figure 13, Figure 14, and Figure 15.

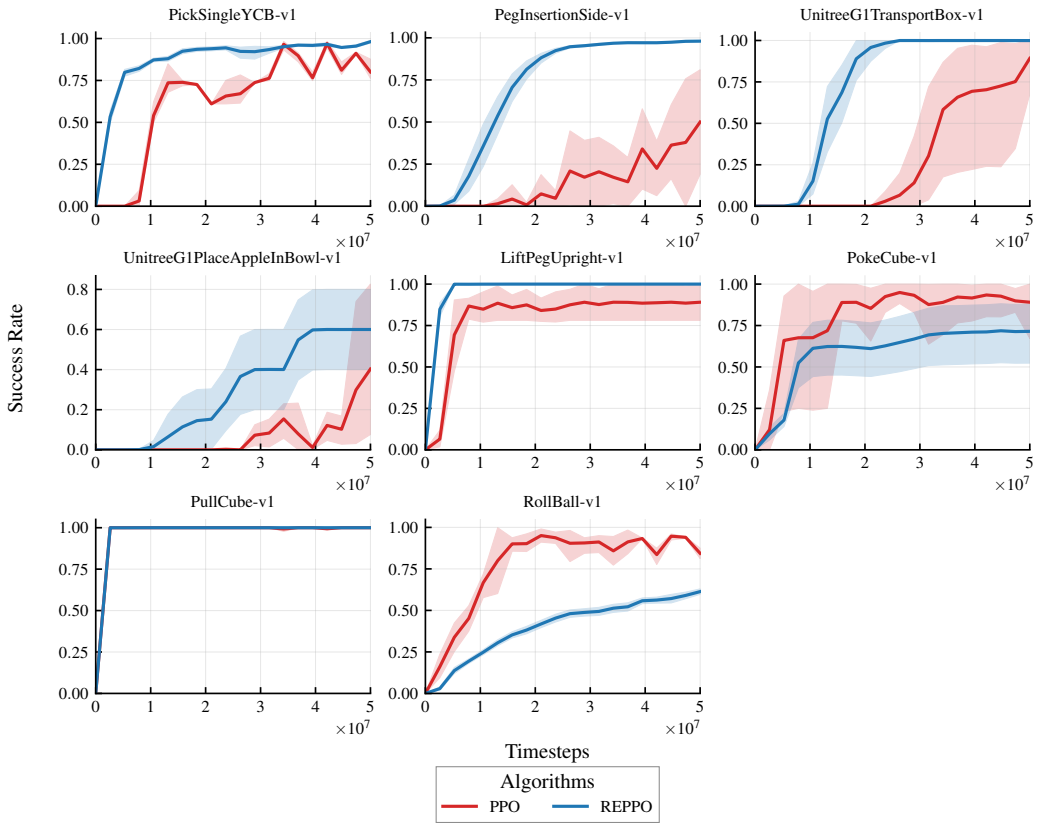


Figure 13: Per-environment results on the ManiSkill suite

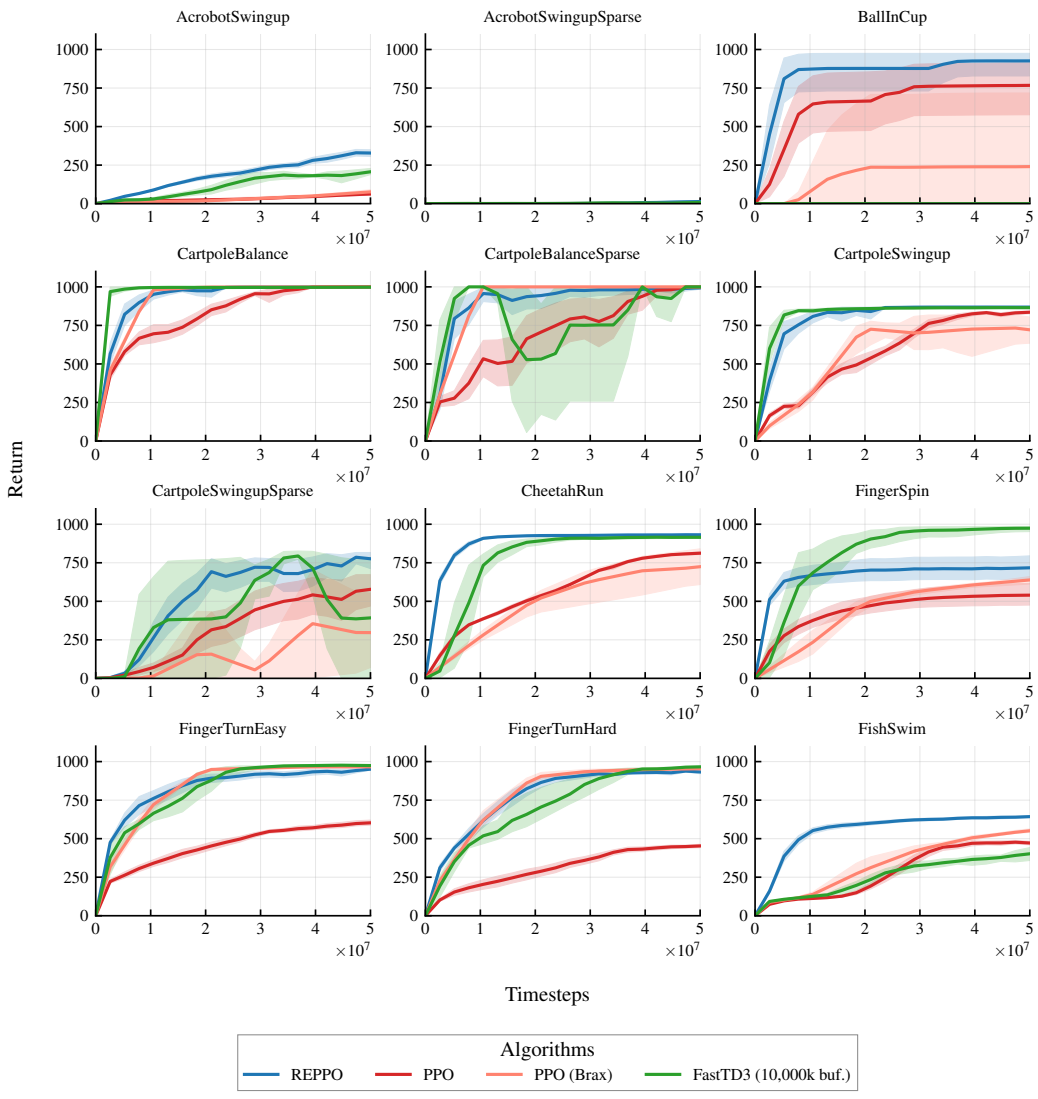


Figure 14: Per-environment results on the mujoco_playground DMC suite

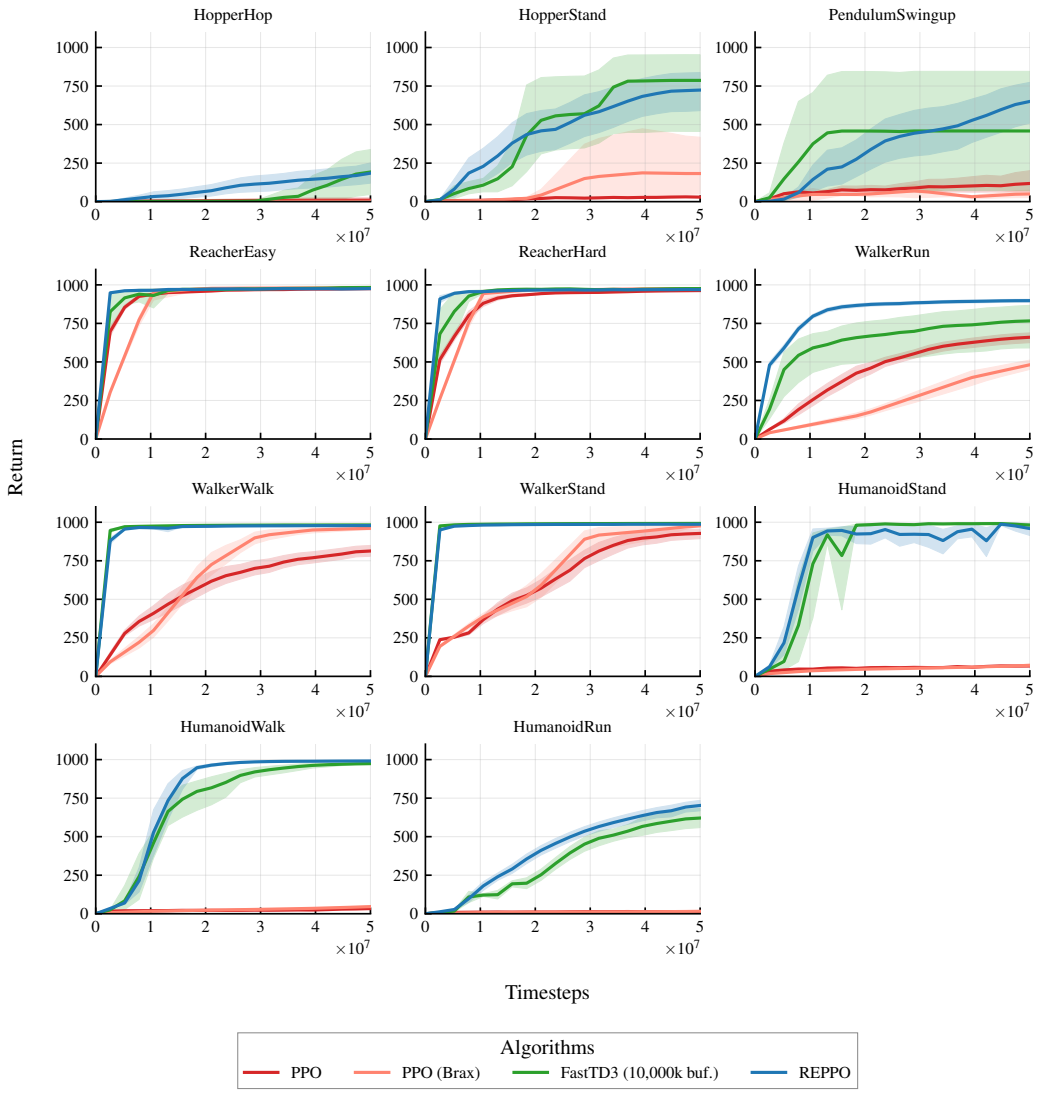


Figure 15: Per-environment results on the mujoco_playground DMC suite

F PSEUDOCODE

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 approximate $V_{t+1} \leftarrow Q_\phi(x_{t+1}, a_{t+1})$ with $a_{t+1} \sim \pi_{\theta'}(\cdot|x_{t+1})$
    Compute $\psi_t \leftarrow f_\phi(x_{t+1}, a_{t+1})$
    // Maximum entropy augmented reward, see Subsection 3.1
    $\tilde{r}_t \leftarrow r_t - \alpha \log \pi_\theta(a_{t+1}|x_{t+1})$
    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 Subsection 3.1
  for $t = T - 1$ *down to* 0 **do**
    $G_t^\lambda \leftarrow \tilde{r}_t + \gamma[(1 - d_t)(\lambda G_{t+1}^\lambda + (1 - \lambda)V_{t+1})]$
  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 Subsection 3.3
      $L_Q \leftarrow \frac{1}{B} \sum \text{CrossEntropy}(Q_\phi(x_i, a_i), \text{Cat}(G_i^\lambda))$
      // Auxiliary task, see Subsection 3.3
      $L_{aux} \leftarrow \frac{1}{B} \sum ||f_\phi(x_i, a_i) - \psi_i||^2]$
      Update critic: $\phi \leftarrow \phi - \alpha_Q \nabla_\phi (L_Q + \beta L_{aux})$
      // Actor update with entropy and KL regularization, see Subsection 3.1 and Subsection 3.2
      Sample action $a_i' \sim \pi_\theta(\cdot|x_i)$
      Sample k actions $\bar{a}_i \sim \pi_{\theta'}(\cdot|x_i)$
      Compute KL divergence: $D_{\text{KL}}(x_i) \leftarrow \sum_{j=1}^k \log \frac{\pi_{\theta'}(\bar{a}_j|x_i)}{\pi_\theta(\bar{a}_j|x_i)}$
      Policy loss: $L_\pi \leftarrow \frac{1}{B} \sum Q_\phi(x_i, a_i') - e^\alpha \log \pi_\theta(a_i'|x_i) - e^\beta D_{\text{KL}}(x_i)$
      (Alternatively, compute clipped objective)
      Update actor: $\theta \leftarrow \theta + \eta_\pi \nabla_\theta L_\pi$
      Entropy α update: $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha e^\alpha (\frac{1}{B} \sum \mathcal{H}[\pi_\theta(x_i)] - \varepsilon_{\mathcal{H}})$
      KL β update: $\beta \leftarrow \beta - \eta_\beta \nabla_\beta e^\beta (\frac{1}{B} \sum D_{KL}(x_i)] - \varepsilon_{\text{KL}})$
    end
  end
  // Behavior Policy Update
  $\theta' \leftarrow \theta$
end
return *Trained policy* π_θ