Generalized Policy Improvement Algorithms with Theoretically Supported Sample Reuse

James Queeney, Ioannis Ch. Paschalidis, *Fellow, IEEE*, and Christos G. Cassandras, *Life Fellow, IEEE*

***Abstract*— We develop a new class of model-free deep reinforcement learning algorithms for data-driven, learning-based control. Our Generalized Policy Improvement algorithms combine the policy improvement guarantees of on-policy methods with the efficiency of sample reuse, addressing a trade-off between two important deployment requirements for real-world control: (i) practical performance guarantees and (ii) data efficiency. We demonstrate the benefits of this new class of algorithms through extensive experimental analysis on a broad range of simulated control tasks.**

***Index Terms*— Policy improvement, policy optimization, reinforcement learning, sample reuse.**

I. Introduction

Model-free deep reinforcement learning (RL) represents a successful framework for data-driven control in systems with unknown or complex dynamics [1], [2], where a learning-based approach to control is necessary. Model-free deep RL algorithms have demonstrated impressive performance on a variety of simulated control tasks in recent years [3], and serve as fundamental building blocks in other RL approaches such as model-based [4]–[6], offline [7]–[9], and safe [10]–[14] control methods.

Despite this success, adoption of deep RL methods has remained limited for high-stakes real-world control, where stable performance is critical and data collection can be both expensive and time-consuming. In order to reliably deploy data-driven control methods in these real-world settings, we require algorithms that (i) provide practical guarantees on performance throughout training and (ii) make efficient use of data collected in the environment. Unfortunately, these represent competing goals in existing model-free deep RL methods.

Off-policy deep RL algorithms achieve data efficiency through the use of a replay buffer during training, which allows samples to be used for multiple policy updates. Typically, the replay buffer stores millions of samples, which enables state-of-the-art performance on popular benchmarks. However, this aggressive form of sample reuse lacks practical performance guarantees, as both the size of the replay buffer and the sampling method for policy updates are treated as hyperparameters. This prevents off-policy deep RL from being a broadly applicable solution for real-world control, and other data-driven methods are required.

On-policy deep RL algorithms, on the other hand, only use data collected under the current policy in order to provide worst-case performance guarantees at every update. By doing so, on-policy methods guarantee *approximate policy improvement* throughout training, which is often a prerequisite for the deployment of data-driven control in real-world systems. In addition, on-policy approaches require significantly less computation and memory compared to off-policy algorithms [15], making them a viable option in settings that preclude the use of large replay buffers. For these reasons, we consider on-policy algorithms as our starting point in this work. However, the requirement of on-policy data results in high sample complexity and slow learning, which limits the effectiveness of these algorithms in practice.

Given our goal of reliable and efficient training, in this work we combine the policy improvement benefits of on-policy methods with the efficiency of sample reuse. This paper extends our prior work in [16], where we generalized the popular on-policy algorithm Proximal Policy Optimization (PPO) [17] to incorporate sample reuse. Compared to [16], our main contributions are as follows:

1) In Section IV, we introduce a generalized trust region update that serves as the foundation for a unified class of *Generalized Policy Improvement (GPI)* algorithms. Our GPI framework extends many popular on-policy methods to incorporate sample reuse.
2) In Section V, we demonstrate how to optimally reuse data from all recent policies. Our *theoretically supported sample reuse* improves the trade-off between batch size and policy update size throughout training compared to on-policy methods, while retaining the same approximate policy improvement guarantees. This is in contrast to the aggressive sample reuse in off-policy methods that lacks practical performance guarantees.
3) In Section VI, we introduce three examples of our GPI framework. In addition to the generalized version of PPO from [16], we also develop generalized versions of the popular on-policy algorithms Trust Region Policy Optimization (TRPO) [18] and On-Policy Maximum a Posteriori Policy Optimization (VMPO) [19].
4) In Section VII, we demonstrate the practical benefits of our GPI algorithms through extensive experimental analysis on a broad range of continuous control benchmarking tasks from the DeepMind Control Suite [20].

II. Related Work

A. On-Policy Policy Improvement Methods

Our work focuses on policy improvement methods, an algorithmic framework that was first introduced by [21] in Conservative Policy Iteration. The policy improvement bounds proposed by [21] were later refined by [18] and [10], making them compatible with the deep RL setting. These advances motivated the development of many popular on-policy policy improvement methods, including TRPO [18] and PPO [17]. The strong empirical performance of TRPO and PPO has been studied in detail [22]–[24], and these algorithms have served as the foundation for many on-policy extensions [10], [25]–[28]. Given the popularity of TRPO and PPO, we consider both of these algorithms when developing our family of GPI algorithms.

This research was partially supported by the NSF under grants ECCS-1931600, DMS-1664644, CNS-1645681, and IIS-1914792, by the ONR under grant N00014-19-1-2571, by the NIH under grants R01 GM135930 and UL54 TR004130, by the DOE under grants DE-AR0001282 and DE-EE0009696, by AFOSR under grant FA9550-19-1-0158, by the MathWorks, and by Boston University.

James Queeney is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA (e-mail: queeney@merl.com). He performed the majority of this work while with the Division of Systems Engineering, Boston University, Boston, MA 02215 USA.

Ioannis Ch. Paschalidis and Christos G. Cassandras are with the Department of Electrical and Computer Engineering and Division of Systems Engineering, Boston University, Boston, MA 02215 USA (e-mail: yannisp@bu.edu; cgc@bu.edu).

Algorithm 1: On-Policy Policy Improvement Algorithms

Input: initial policy π_0; TV distance trust region parameter ϵ; batch size N.

for $k = 0, 1, 2, \ldots$ **do**

Collect N samples via trajectory rollouts $\tau \sim \pi_k$.

Use N samples from π_k to approximate the expectations in (2).

Update policy by approximately solving the optimization problem in (2). Implementation varies by algorithm.

end

In recent years, on-policy methods have also considered non-parametric target policies obtained by solving policy optimization problems similar to those used in TRPO and PPO [19], [29]. These target policies are then projected back onto the space of parameterized policies. VMPO [19] considers a target policy induced by a reverse Kullback-Leibler (KL) divergence constraint, and its practical trust region implementation can be interpreted from a policy improvement perspective. Therefore, we also propose a generalized version of VMPO as a representative instance of policy improvement algorithms based on non-parametric target policies.

B. Sample Efficiency with Off-Policy Data

The main drawback of on-policy policy improvement algorithms is their requirement of on-policy data, which can lead to slow and inefficient learning. Popular off-policy deep RL algorithms [30]–[33] address this issue by storing data in a large replay buffer and aggressively reusing samples for many policy updates. Typically, these algorithms do not explicitly control the bias introduced by off-policy data, and as a result do not provide practical policy improvement guarantees.

Methods have been proposed to address the bias introduced by off-policy data. One line of work combines on-policy and off-policy updates to address this concern [34]–[39], and some of these methods consider regularization terms that are similar to the generalized KL divergence trust regions in our Generalized TRPO and VMPO algorithms. Other approaches control the bias from off-policy data by modifying the use of the replay buffer, such as ignoring samples whose actions are not likely under the current policy [40] or emphasizing more recent experience [41]. The theoretically supported sample reuse that we consider in our GPI algorithms can be viewed as restricting the size of the replay buffer to only contain recent policies, as well as applying a non-uniform weighting to the samples.

III. Preliminaries

A. Reinforcement Learning Framework

We represent the sequential decision making problem as an infinite-horizon, discounted Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ is the transition probability function where $P(\mathcal{S})$ represents the space of probability measures over $\mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 \in P(\mathcal{S})$ is the initial state distribution, and γ is the discount rate.

We model the agent's decisions as a stationary policy $\pi_{\boldsymbol{\theta}} : \mathcal{S} \to P(\mathcal{A})$ parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. In this work, we consider policies parameterized by neural networks where $\pi_{\boldsymbol{\theta}}(\cdot \mid s)$ shares the same support for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. This is true of most popular policy parameterizations in deep RL, including the multivariate Gaussian policy we consider in our experiments. In the remainder of the paper, we write π without the subscript for ease of notation.

Our goal is to find a policy π that maximizes the expected total discounted return $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, where $\tau \sim \pi$ represents a trajectory sampled according to $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. We denote the state value function of π as $V^\pi(s) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right]$, the state-action value function as $Q^\pi(s, a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\right]$, and the advantage function as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. A policy π also induces a normalized discounted state visitation distribution d^π, where $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s \mid \rho_0, \pi, p)$. We write the corresponding normalized discounted state-action visitation distribution as $d^\pi(s, a) = d^\pi(s)\pi(a \mid s)$, where we make it clear from the context whether d^π refers to a distribution over states or state-action pairs.

B. On-Policy Policy Improvement Algorithms

Many popular on-policy algorithms can be interpreted as approximately maximizing the following policy improvement lower bound, which was first developed by [21] and later refined by [18] and [10].

Theorem 1 (From [10]): Consider any policy π and a current policy π_k. Then, we have that

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s, a)\right] - \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{TV}\left(\pi, \pi_k\right)(s)\right], \quad (1)$$

where $\mathrm{TV}\left(\pi, \pi_k\right)(s) = \frac{1}{2}\int_{\mathcal{A}} |\pi(a \mid s) - \pi_k(a \mid s)| \, \mathrm{d}a$ represents the Total Variation (TV) distance between the distributions $\pi(\cdot \mid s)$ and $\pi_k(\cdot \mid s)$, and $C^{\pi,\pi_k} = \max_{s\in\mathcal{S}} \left|\mathbb{E}_{a\sim\pi(\cdot\,|s)}\left[A^{\pi_k}(s, a)\right]\right|$.

We refer to the first term on the right-hand side of (1) as the surrogate objective and the second term as the penalty term. Rather than directly maximize the lower bound in Theorem 1, on-policy policy improvement algorithms typically maximize the surrogate objective while bounding the risk of each policy update via a constraint on the penalty term. This leads to updates with the following form.

Definition 1: For a given choice of trust region parameter $\epsilon > 0$, the *on-policy trust region update* has the form

$$\begin{aligned} \pi_{k+1} = \arg\max_{\pi} & \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s, a)\right] \\ \text{s.t.} & \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{TV}\left(\pi, \pi_k\right)(s)\right] \leq \frac{\epsilon}{2}. \end{aligned} \quad (2)$$

By maximizing the surrogate objective from (1) while constraining the magnitude of the penalty term from (1), the trust region update in (2) limits the *worst-case performance decline* at every update to

$$J(\pi_{k+1}) - J(\pi_k) \geq -\frac{\epsilon\gamma C^{\pi_{k+1},\pi_k}}{(1-\gamma)^2}. \quad (3)$$

Therefore, we say that on-policy algorithms based on the trust region update in (2) deliver *approximate* policy improvement guarantees. In addition, practical deep RL implementations of this update introduce additional approximations through the use of sample-based estimates.

The high-level framework of on-policy policy improvement algorithms is described in Algorithm 1. The difference between popular on-policy algorithms is primarily due to how they approximately solve the optimization problem in (2). We provide details for PPO, TRPO, and VMPO in Section VI.

IV. Generalized Trust Region Update

The expectations that appear in the surrogate objective and penalty term of (1) can be estimated using samples collected under the current policy π_k. Therefore, this bound motivates algorithms that are practical to implement, but may result in slow learning due to their requirement of on-policy data. The goal of our work is to improve the efficiency of on-policy algorithms through sample reuse, without sacrificing their approximate policy improvement guarantees. In order to relax the on-policy requirement of the expectations that appear in (1), we leverage our Generalized Policy Improvement lower bound from [16].

Theorem 2 (From [16]): Consider any policy π and M prior policies π_{k-i}, $i = 0, \dots, M-1$, where $k \geq M-1$. Let ν be any choice of mixture distribution over these M prior policies, where $0 \leq \nu_i \leq 1$ is the probability assigned to π_{k-i}, $\sum_{i=0}^{M-1} \nu_i = 1$, and $\mathbb{E}_{i\sim\nu}[\cdot]$ represents an expectation determined by this mixture distribution. Then, we have that

$$J(\pi) - J(\pi_k) \geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] - \frac{2\gamma C^{\pi,\pi_k}}{(1-\gamma)^2} \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k-i}\right)(s)\right]\right], \quad (4)$$

where C^{π,π_k} is defined as in Theorem 1.

The expectations that appear in our Generalized Policy Improvement lower bound can be estimated using a mixture of samples collected under prior policies, so Theorem 2 provides insight into how we can reuse samples while still providing guarantees on performance throughout training. The cost of sample reuse is that the penalty term now depends on the expected TV distance between the new policy and our prior policies, rather than the current policy. Note that we recover the on-policy lower bound when ν is chosen to place all weight on the current policy, so Theorem 1 is a special case of Theorem 2.

Because the structure of our generalized lower bound remains the same as the on-policy lower bound, we can use the same techniques to motivate practical policy improvement algorithms. Just as the on-policy lower bound motivated the policy update in (2), we can use Theorem 2 to motivate a generalized policy update of the form

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k-i}\right)(s)\right]\right] \leq \frac{\epsilon}{2}, \quad (5)$$

where ϵ represents the same trust region parameter used in the on-policy case. By the triangle inequality of TV distance, we have that

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k-i}\right)(s)\right]\right] \leq \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k}\right)(s)\right]\right] + \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_k, \pi_{k-i}\right)(s)\right]\right], \quad (6)$$

where the first term on the right-hand side is the expected one-step TV distance and the second term does not depend on π. Therefore, we can satisfy the trust region in (5) by controlling the expected one-step TV distance, which is often easier to work with in practice. For an appropriate choice of ϵ_{GPI}, this leads to the following policy update.

Definition 2: For a given choice of trust region parameter $\epsilon_{\text{GPI}} > 0$, the *generalized trust region update* has the form

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k}\right)(s)\right]\right] \leq \frac{\epsilon_{\text{GPI}}}{2}. \quad (7)$$

Similar to the on-policy trust region update in (2), the generalized trust region update in (7) also provides approximate policy improvement guarantees due to its connection to the Generalized Policy Improvement lower bound in Theorem 2. In order to deliver these approximate policy improvement guarantees, the generalized update still depends on the advantage function with respect to the *current* policy π_k, which must be approximated using off-policy estimation techniques in practice.

Next, we describe how to select the generalized trust region parameter ϵ_{GPI} and mixture distribution ν over prior policies in order to provide guarantees on the risk of every policy update while optimizing key quantities of interest. Principled choices of ϵ_{GPI} and ν result in *theoretically supported sample reuse*.

V. Theoretically Supported Sample Reuse

A. Generalized Trust Region Parameter

From (6), we see that the generalized update in Definition 2 satisfies the trust region in (5) for any ϵ_{GPI} such that

$$\frac{\epsilon_{\text{GPI}}}{2} \leq \frac{\epsilon}{2} - \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_k, \pi_{k-i}\right)(s)\right]\right]. \quad (8)$$

While the adaptive choice of ϵ_{GPI} given by (8) will successfully control the magnitude of the penalty term in (4), it only indirectly provides insight into how ϵ_{GPI} depends on the choice of ν. In order to establish a direct connection between ϵ_{GPI} and ν, our analysis considers a slightly stronger trust region update given by

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a)\right]\right] \quad \text{s.t.} \quad \mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi, \pi_{k}\right)(s)\right] \leq \frac{\epsilon_{\text{GPI}}}{2}, \quad i = 0, \dots, M-1. \quad (9)$$

Theorem 3: Consider policy updates determined by (9), where

$$\epsilon_{\text{GPI}} = \frac{\epsilon}{\mathbb{E}_{i\sim\nu}[i+1]}. \quad (10)$$

Then, we have that π_{k+1} satisfies

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_{k+1}, \pi_{k-i}\right)(s)\right]\right] \leq \frac{\epsilon}{2}, \quad (11)$$

so that π_{k+1} is a feasible solution to (5).

Proof: Using the triangle inequality for TV distance, we have

$$\mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_{k+1}, \pi_{k-i}\right)(s)\right]\right] \leq \mathop{\mathbb{E}}_{i\sim\nu}\left[\sum_{j=0}^{i} \mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\mathrm{TV}\left(\pi_{k-j+1}, \pi_{k-j}\right)(s)\right]\right] \leq \mathop{\mathbb{E}}_{i\sim\nu}\left[\frac{\epsilon_{\text{GPI}}}{2}\cdot(i+1)\right] = \frac{\epsilon_{\text{GPI}}}{2}\cdot\mathop{\mathbb{E}}_{i\sim\nu}[i+1] = \frac{\epsilon}{2},$$

where we have used the trust region constraints in (9) from all prior updates to bound each term inside the summation by $\epsilon_{\text{GPI}}/2$. ∎

Importantly, Theorem 3 implies that the policy update in (9) has the same worst-case performance guarantee given by (3) as the on-policy update in (2). In practice, we consider the generalized trust

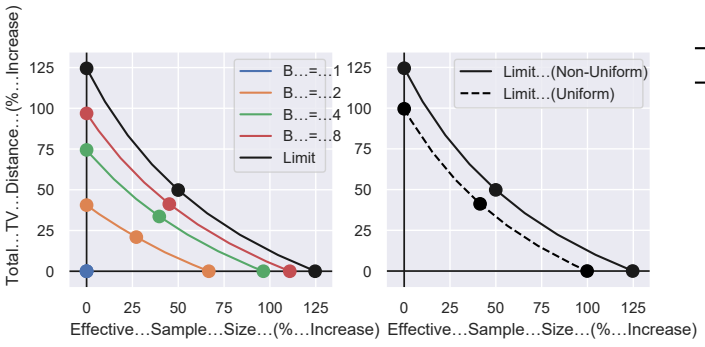

Fig. 1. Benefit of GPI update compared to on-policy case. Represents percent increases in effective sample size and total TV distance update size for all possible values of $\boldsymbol{\kappa} \in [\mathbf{0}, \mathbf{1}]$. Markers indicate $\boldsymbol{\kappa} = \mathbf{0.0}, \mathbf{0.5}, \mathbf{1.0}$. Left: Comparison across several values of $\boldsymbol{B}$. Right: Comparison of non-uniform and uniform weights for large $\boldsymbol{B}$.

region update in (7), which relaxes the trust region in (9). We find that the generalized update in (7) also satisfies (11) in practice, and tends to be conservative because ϵ_{GPI} in (10) is based on applying the triangle inequality between every prior policy. In addition, practical implementations based on clipping mechanisms or backtracking line searches can ensure that (11) holds.

The form of ϵ_{GPI} in (10) clearly demonstrates the trade-off of sample reuse. As we reuse older data, we must consider smaller one-step trust regions at every policy update in order to guarantee the same level of risk.

B. Mixture Distribution

Despite the need for smaller one-step trust regions at every policy update, we can show that our generalized policy update improves key quantities of interest when the mixture distribution ν is chosen in a principled manner. Note that $\nu_i > 0$ indicates that data from π_{k-i} will be used during updates, so the choice of ν determines how many prior policies to consider in addition to how to weight their contributions.

We write the on-policy sample size as $N = Bn$, where n represents the smallest possible batch size for data collection (e.g., the length of one full trajectory) and B is a positive integer. On-policy policy improvement algorithms update the policy according to (2) after every N samples collected, where the expectations are approximated using empirical averages calculated with these N samples. For the generalized case, we can combine data across several prior policies to construct the batch used for calculating the generalized policy update in (7). Therefore, sample reuse allows us to make policy updates after every n samples collected.

It is common to consider auxiliary metrics to inform sampling schemes in deep RL [42], [43]. In order to determine ν, we focus on two important quantities in policy optimization: (i) effective sample size and (ii) total TV distance update size. We now define these metrics for our generalized policy updates.

Definition 3: The *effective sample size* used for generalized policy updates is given by

$$f_{\text{ESS}}(\nu) = \frac{n}{\sum_{i=0}^{M-1} \nu_i^2},$$

where ν_i represents the probability assigned to data collected under π_{k-i}, compared to $N = Bn$ in on-policy algorithms.

Definition 4: The *total TV distance update size* of generalized policy updates for every N samples collected is given by

$$f_{\text{TV}}(\nu) = B \cdot \frac{\epsilon_{\text{GPI}}}{2} = \frac{B}{\sum_{i=0}^{M-1} \nu_i \, (i+1)} \cdot \frac{\epsilon}{2},$$

Algorithm 2: Generalized Policy Improvement Algorithms

Input: initial policy π_0; TV distance trust region parameter ϵ; on-policy batch size $N = Bn$, where n represents minimum batch size; trade-off parameter κ.

Calculate mixture distribution ν using (12), and let M be the number of prior policies with non-zero weighting.

Calculate generalized trust region parameter ϵ_{GPI} using (10).

for $k = 0, 1, 2, \ldots$ **do**

Collect n samples via trajectory rollouts $\tau \sim \pi_k$.

Use n samples from each of π_{k-i}, $i = 0, \ldots, M-1$, to approximate the expectations in (7).

Update policy by approximately solving the optimization problem in (7). Implementation varies by algorithm.

end

compared to $\epsilon/2$ in on-policy algorithms.

The effective sample size [44] adjusts the sample size to account for increased variance due to non-uniform weights, and reduces to the standard sample size definition of Mn for the case of uniform weights over the last M policies. A larger effective sample size results in more accurate estimates of the expectations that we must approximate in policy updates, and leads to a more diverse batch of data which can be useful when reward signals are sparse [45]. A larger total TV distance update size, on the other hand, allows for more aggressive exploitation of the available information, which can lead to faster learning throughout training. Although optimizing these metrics does not guarantee improved performance compared to on-policy algorithms, we see in Section VII that experimentally this is often the case.

Using the following result, we select ν to optimize $f_{\text{ESS}}(\nu)$ and $f_{\text{TV}}(\nu)$ relative to the on-policy case.

Theorem 4: Fix the trade-off parameter $\kappa \in [0, 1]$, and select the mixture distribution ν according to the convex optimization problem

$$\begin{aligned} \nu^*(\kappa) = \arg\min_{\nu} \; & \kappa \cdot \frac{\sum_{i=0}^{\bar{M}-1} \nu_i^2}{c_{\text{ESS}}} + (1-\kappa) \cdot \frac{\sum_{i=0}^{\bar{M}-1} \nu_i \, (i+1)}{c_{\text{TV}}} \\ \text{s.t.} \; & \sum_{i=0}^{\bar{M}-1} \nu_i^2 \leq \frac{1}{B}, \quad \sum_{i=0}^{\bar{M}-1} \nu_i \, (i+1) \leq B, \\ & \sum_{i=0}^{\bar{M}-1} \nu_i = 1, \; \nu_i \geq 0, \; i = 0, \ldots, \bar{M}-1, \end{aligned} \tag{12}$$

where $c_{\text{ESS}}, c_{\text{TV}} \geq 0$ are scaling coefficients and $\bar{M}$ is large. Then, we have that $f_{\text{ESS}}(\nu^*) \geq Bn$ and $f_{\text{TV}}(\nu^*) \geq \epsilon/2$.

Proof: Note that $f_{\text{ESS}}(\nu)$ and $f_{\text{TV}}(\nu)$ only depend on ν in their denominators. Therefore, we can maximize these quantities by minimizing the quantities in their denominators that depend on ν. By considering a convex combination determined by the trade-off parameter κ and applying scaling coefficients, we arrive at the objective in (12).

Next, we consider the constraints in (12). The first constraint implies $f_{\text{ESS}}(\nu) \geq Bn$ and the second constraint implies $f_{\text{TV}}(\nu) \geq \epsilon/2$. Therefore, these constraints guarantee that the effective sample size and total TV distance update size are at least as large as in the on-policy case. The remaining constraints ensure that ν is a distribution.

Finally, note that $\nu_i = 1/M$, $i = 0, \ldots, M-1$, is a feasible solution to (12) for $B \leq M \leq 2B - 1$. Therefore, (12) is a feasible optimization problem. ∎

In the left-hand side of Fig. 1, we see the benefits of using the optimal mixture distribution from Theorem 4 for different values of B. As B becomes larger, on-policy algorithms require larger

batch sizes and the benefit of our generalized approach increases. In the right-hand side of Fig. 1, we see that the main benefit of optimizing ν comes from determining the appropriate number of prior policies to consider, with non-uniform weights offering additional improvements. Together, Theorem 3 and Theorem 4 provide theoretical support for the sample reuse we consider in this work.

VI. Generalized Policy Improvement Algorithms

The theory that we have developed can be used to construct generalized versions of on-policy algorithms with theoretically supported sample reuse. We refer to this class of algorithms as Generalized Policy Improvement (GPI) algorithms. The high-level framework for these algorithms is shown in Algorithm 2. By using the optimal mixture distribution from Theorem 4, GPI algorithms only require modest increases in memory and computation compared to on-policy algorithms. In addition, due to their use of one-step trust regions, GPI algorithms only need access to the current policy and value function in order to compute updates. As a result, deep RL implementations do not require storage of any additional neural networks compared to on-policy algorithms.

In this section, we provide details on three algorithms from this class: Generalized PPO (GePPO), Generalized TRPO (GeTRPO), and Generalized VMPO (GeVMPO).

A. Generalized PPO

PPO approximates the on-policy trust region update in (2) using the policy update

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\min \left(\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a), \right. \right. \\ \left. \left. \operatorname{clip}\left(\frac{\pi(a \mid s)}{\pi_k(a \mid s)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_k}(s,a) \right) \right], \quad (13)$$

where $\operatorname{clip}(x,l,u) = \min(\max(x,l),u)$ and the maximization is implemented using minibatch stochastic gradient ascent. The policy update in (13) approximately maximizes a lower bound on the on-policy surrogate objective, while also heuristically enforcing the on-policy trust region in (2) through the use of the clipping mechanism

$$\operatorname{clip}\left(\frac{\pi(a \mid s)}{\pi_k(a \mid s)}, 1-\epsilon, 1+\epsilon \right)$$

in the second term [16]. The clipping mechanism accomplishes this by removing any incentive for the probability ratio to deviate more than ϵ from its starting point during every policy update.

In order to develop a generalized version of PPO that approximates the generalized trust region update in (7), we desire a clipping mechanism that heuristically enforces the generalized trust region in (7). As shown in [16], the clipping mechanism

$$\operatorname{clip}\left(\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)}, \frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)} - \epsilon_{\text{GPI}}, \frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)} + \epsilon_{\text{GPI}} \right)$$

accomplishes this goal by removing the incentive for the probability ratio to deviate more than ϵ_{GPI} from its starting point of ${\pi_k(a|s)}/{\pi_{k-i}(a|s)}$. By applying this generalized clipping mechanism and considering a lower bound on the generalized surrogate objective, we arrive at the Generalized PPO (GePPO) update

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{i\sim\nu} \left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}} \left[\right. \right. \\ \min \left(\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a), \operatorname{clip}\left(\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)}, \right. \right. \\ \left. \left. \left. \left. \frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)} - \epsilon_{\text{GPI}}, \frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)} + \epsilon_{\text{GPI}} \right) A^{\pi_k}(s,a) \right) \right] \right]. \quad (14)$$

See [16] for additional details, where we first introduced GePPO.

B. Generalized TRPO

TRPO approximates the on-policy update in (2) by instead applying a forward KL divergence trust region, leading to the policy update

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a) \right] \\ \text{s.t.} \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{KL}\left(\pi_k \| \pi\right)(s) \right] \leq \delta, \quad (15)$$

where $\mathrm{KL}\left(\pi_k \| \pi\right)(s)$ represents the forward KL divergence of the distribution $\pi(\cdot \mid s)$ from the distribution $\pi_k(\cdot \mid s)$. For $\delta = \epsilon^2/2$, an application of Pinsker's inequality [46] followed by Jensen's inequality shows that

$$\mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{TV}\left(\pi, \pi_k\right)(s) \right] \leq \sqrt{\frac{1}{2} \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{KL}\left(\pi_k \| \pi\right)(s) \right]} \\ \leq \sqrt{\frac{\delta}{2}} = \frac{\epsilon}{2},$$

so the update in (15) satisfies the trust region in (2).

TRPO considers a first-order expansion of the surrogate objective and second-order expansion of the forward KL divergence trust region in (15), leading to the approximations

$$\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}} \left[\frac{\pi(a \mid s)}{\pi_k(a \mid s)} A^{\pi_k}(s,a) \right] \approx \mathbf{g}_k' \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right), \\ \mathop{\mathbb{E}}_{s\sim d^{\pi_k}} \left[\mathrm{KL}\left(\pi_k \| \pi\right)(s) \right] \approx \frac{1}{2} \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right)' \mathbf{F}_k \left(\boldsymbol{\theta} - \boldsymbol{\theta}_k \right),$$

where $\boldsymbol{\theta}, \boldsymbol{\theta}_k \in \boldsymbol{\Theta}$ are the parameterizations of π and π_k, respectively. Using these approximations, the TRPO policy update admits a closed-form solution for the parameterization of π_{k+1} given by $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \beta \mathbf{v}$, where $\mathbf{v} = \mathbf{F}_k^{-1} \mathbf{g}_k$ is the update direction and $\beta = \sqrt{2\delta / \mathbf{v}' \mathbf{F}_k \mathbf{v}}$. Note that the update direction cannot be calculated directly in high dimensions, so instead it is approximated by applying a finite number of conjugate gradient steps to $\mathbf{F}_k \mathbf{v} = \mathbf{g}_k$. Finally, a backtracking line search is performed to guarantee that the trust region in (15) is satisfied.

It is straightforward to extend TRPO to the generalized setting. We approximate the generalized update in (7) by instead applying a forward KL divergence trust region, leading to the policy update

$$\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{i\sim\nu} \left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}} \left[\frac{\pi(a \mid s)}{\pi_{k-i}(a \mid s)} A^{\pi_k}(s,a) \right] \right] \\ \text{s.t.} \mathop{\mathbb{E}}_{i\sim\nu} \left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}} \left[\mathrm{KL}\left(\pi_k \| \pi\right)(s) \right] \right] \leq \delta_{\text{GPI}} \quad (16)$$

with $\delta_{\text{GPI}} = \epsilon_{\text{GPI}}^2/2$. We consider the same approximations and optimization procedure as TRPO to implement the Generalized TRPO (GeTRPO) update.

C. Generalized VMPO

VMPO approximates the on-policy update in (2) by instead considering a reverse KL divergence trust region, and treats the state-action visitation distribution as the variable rather than the policy. We write the new and current state-action visitation distributions ψ, ψ_k as

$$\psi(s,a) = d^{\pi_k}(s) \pi(a \mid s), \quad \psi_k(s,a) = d^{\pi_k}(s) \pi_k(a \mid s),$$

which results in the non-parametric VMPO update

$$\psi_{\text{targ}} = \arg\max_{\psi} \mathop{\mathbb{E}}_{(s,a)\sim\psi} \left[A^{\pi_k}(s,a) \right] \\ \text{s.t. } \mathrm{KL}\left(\psi \| \psi_k\right) \leq \delta. \quad (17)$$

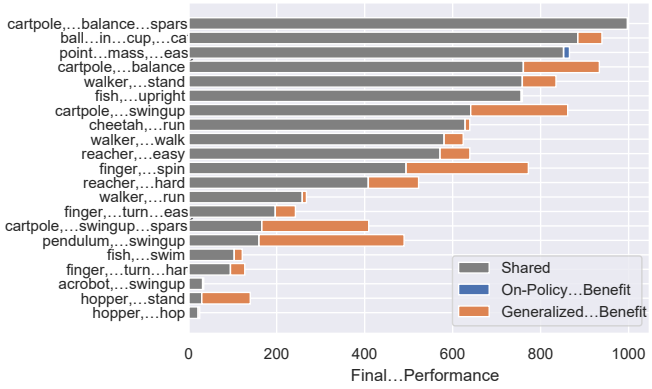

Fig. 2. Generalized vs. on-policy final performance by task. Bars represent final performance of the best performing GPI algorithm and the best performing on-policy algorithm. Excludes 7 tasks where no learning occurs under any algorithm. Sorted from high to low based on on-policy performance.

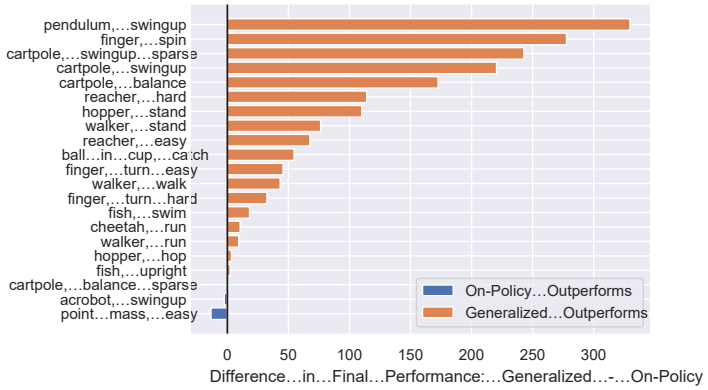

Fig. 3. Difference between generalized and on-policy final performance by task. Bars represent difference in final performance between the best performing GPI algorithm and the best performing on-policy algorithm. Excludes 7 tasks where no learning occurs under any algorithm. Sorted from high to low.

As shown in [19], the target distribution can be written as

$$\psi_{\text{targ}}(s,a) = d^{\pi_k}(s)\pi_k(a \mid s)w(s,a),$$

where

$$w(s,a) = \exp\left(A^{\pi_k}(s,a)/\lambda^*\right) \,/\, Z(\lambda^*),$$
$$Z(\lambda^*) = \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}}\left[\exp\left(A^{\pi_k}(s,a)/\lambda^*\right)\right],$$

and

$$\lambda^* = \arg\min_{\lambda\geq 0}\ \lambda\delta + \lambda\log\left(Z(\lambda)\right)$$

is the optimal solution to the corresponding dual problem.

Next, VMPO projects this target distribution back onto the space of parametric policies, while guaranteeing that the new policy satisfies a forward KL divergence trust region. Therefore, VMPO guarantees approximate policy improvement in both the initial non-parametric step and the subsequent projection step. This results in the constrained maximum likelihood update

$$\begin{aligned}\pi_{k+1} = \arg\max_{\pi}\ & \mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_k}}\left[w(s,a)\log\pi(a \mid s)\right] \\ \text{s.t.}\ & \mathop{\mathbb{E}}_{s\sim d^{\pi_k}}\left[\text{KL}\left(\pi_k \| \pi\right)(s)\right] \leq \delta,\end{aligned} \tag{18}$$

which we implement by applying the same approximation techniques used in TRPO.

In order to generalize VMPO, we begin by approximating the generalized update in (7) with a reverse KL divergence trust region, and we transform the update to consider state-action visitation distributions given by

$$\psi(s,a) = \mathop{\mathbb{E}}_{i\sim\nu}\left[d^{\pi_{k-i}}(s)\right]\pi(a \mid s),$$
$$\psi_k(s,a) = \mathop{\mathbb{E}}_{i\sim\nu}\left[d^{\pi_{k-i}}(s)\right]\pi_k(a \mid s).$$

Using these generalized visitation distributions, the non-parametric update has the same form as (17) with δ replaced by δ_{GPI}. This results in the target distribution

$$\psi_{\text{targ}}(s,a) = \mathop{\mathbb{E}}_{i\sim\nu}\left[d^{\pi_{k-i}}(s)\right]\pi_k(a \mid s)w(s,a),$$

where $w(s,a)$ has the same form as in the on-policy case, the normalizing coefficient is now given by

$$Z(\lambda^*) = \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)}\exp\left(A^{\pi_k}(s,a)/\lambda^*\right)\right]\right],$$

and λ^* is the optimal solution to the corresponding dual problem as in the on-policy case.

The projection step in the generalized case considers a forward KL divergence trust region, resulting in the Generalized VMPO (GeVMPO) update

$$\begin{aligned}\pi_{k+1} = \arg\max_{\pi}\ & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{(s,a)\sim d^{\pi_{k-i}}}\left[\right.\right. \\ & \left.\left.\frac{\pi_k(a \mid s)}{\pi_{k-i}(a \mid s)}w(s,a)\log\pi(a \mid s)\right]\right] \\ \text{s.t.}\ & \mathop{\mathbb{E}}_{i\sim\nu}\left[\mathop{\mathbb{E}}_{s\sim d^{\pi_{k-i}}}\left[\text{KL}\left(\pi_k \| \pi\right)(s)\right]\right] \leq \delta_{\text{GPI}}.\end{aligned} \tag{19}$$

VII. Experiments

In order to analyze the performance of our GPI algorithms, we consider the full set of 28 continuous control benchmark tasks in the DeepMind Control Suite [20]. This benchmark set covers a broad range of continuous control tasks, including a variety of classic control, goal-oriented manipulation, and locomotion tasks. In addition, the benchmark tasks vary in complexity, both in terms of dimensionality and sparsity of reward signals. Finally, each task has a horizon length of 1,000 and $r(s,a) \in [0,1]$ for every state-action pair, resulting in a total return between 0 and 1,000.

We focus our analysis on the comparison between GPI algorithms and their on-policy policy improvement counterparts. Note that we do not claim state-of-the-art performance, but instead we are interested in evaluating the benefits of theoretically supported sample reuse in the context of policy improvement algorithms. By doing so, we can support the use of GPI algorithms in settings where on-policy methods are currently the most viable option for data-driven control.

A. Implementation Details

In our experiments, we follow [16] by considering default network architectures and hyperparameters commonly found in the literature [22]–[24]. We model the policy π as a multivariate Gaussian distribution with diagonal covariance, where the mean action for a given state is parameterized by a neural network with two hidden layers of 64 units each and tanh activations. The state-independent standard deviation of each action dimension is parameterized separately. We represent the value function using a neural network with the same structure. For each task, we train our policy for a total of 1 million

TABLE I
TASK CLASSIFICATION BY ALGORITHM

Task Classification	PPO	TRPO	VMPO	Best
On-Policy Outperforms	3	1	3	2
Generalized Outperforms	18	19	17	19
No Learning	7	8	8	7
Total Number of Tasks	28	28	28	28

steps, and we average over 5 random seeds. Note that some of the more difficult, high-dimensional tasks do not demonstrate meaningful learning under these default choices for any of the policy improvement algorithms we consider. These tasks likely require significantly longer training, different network architectures, or other algorithmic frameworks to successfully learn.

We evaluate performance across three on-policy policy improvement algorithms: PPO, TRPO, and VMPO. For these on-policy algorithms, we consider the default batch size of $N = 2{,}048$, which we write as $B = 2$ and $n = 1{,}024$ since every task we consider has a horizon length of 1,000. We consider $\epsilon = 0.2$ for the TV distance trust region parameter in (2), which corresponds to the clipping parameter ϵ in PPO. We calculate the KL divergence trust region parameter as $\delta = \epsilon^2/2 = 0.02$. We estimate advantages using Generalized Advantage Estimation (GAE) [47].

We also consider generalized versions of each on-policy algorithm: GePPO, GeTRPO, and GeVMPO. When selecting the mixture distribution over prior policies according to Theorem 4, we consider the trade-off parameter with the best final performance from the set of $\kappa = 0.0, 0.5, 1.0$. For $B = 2$, these choices result in using data from the prior 3 or 4 policies. The generalized trust region parameter ϵ_{GPI} is calculated according to Theorem 3, and we set $\delta_{\text{GPI}} = \epsilon^2_{\text{GPI}}/2$. As in the on-policy case, our GPI algorithms require estimates of $A^{\pi_k}(s, a)$. In order to accomplish this, we consider an off-policy variant of GAE that uses the V-trace value function estimator [48]. See [16] for additional implementation details.[1]

B. Overview of Experimental Results

In order to evaluate the benefits of our GPI framework, we compare the best performing on-policy algorithm to the best performing GPI algorithm for every task where learning occurs. Fig. 2 shows the final performance of these algorithms by task, and Fig. 3 shows the difference in final performance by task. From these results, we see a clear performance gain from our generalized approach. Out of 21 tasks where learning occurs, our GPI algorithms outperform on-policy algorithms in 19 tasks. In the 2 tasks where on-policy algorithms outperform, the performance difference is small. On the other hand, in the tasks where our GPI algorithms outperform, we often observe a significant performance difference. We see an improvement of more than 50% from our GPI algorithms in 4 of the tasks and an improvement of more than 10% in 13 of the tasks. Note that we also observe similar trends when comparing any on-policy algorithm to its corresponding generalized version, as summarized in Table I.

C. Analysis of Sparse Reward Tasks

From Fig. 3, it is clear that our GPI framework results in improved performance across a broad range of tasks. We find that this benefit is most pronounced for tasks with sparse reward signals that are difficult to find and exploit. In order to quantify the sparsity of every task in the benchmarking set, we measure the percentage of samples that contain a negligible reward signal under a random policy. We measure this statistic by collecting 100,000 samples under a random policy, and we calculate the percentage of samples where $r(s, a) \leq 0.01$.

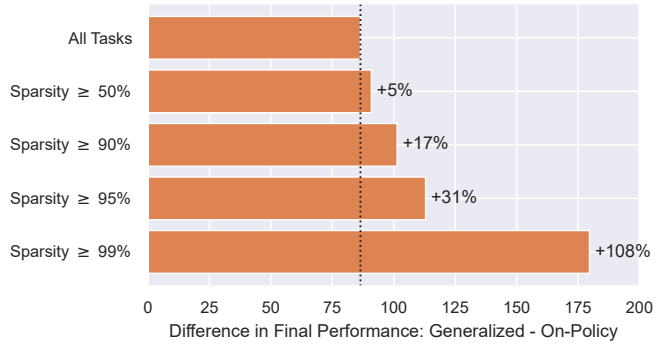

Fig. 4. Average difference between generalized and on-policy final performance for different sparsity levels. Bars represent difference in final performance between the best performing GPI algorithm and the best performing on-policy algorithm, averaged across all tasks with the specified sparsity level. Excludes 7 tasks where no learning occurs under any algorithm. Labels represent improvement relative to the average performance gain across all tasks, denoted by the vertical dotted line.

Fig. 4 shows the average performance gain of our GPI algorithms across tasks, where we have considered different levels of task sparsity. As we focus on tasks with increasingly sparse reward signals (i.e., a larger sparsity metric), we see that the benefits of our GPI framework also increase. For extremely sparse reward tasks where a random policy receives a reward signal less than 1% of the time (i.e., sparsity $\geq 99\%$), the average performance gain from our GPI framework more than doubles relative to the average across all tasks. In fact, this set of extremely sparse reward tasks includes the 3 tasks where our GPI algorithms demonstrate the largest total gain in performance and the 4 tasks where our GPI algorithms demonstrate the largest percentage gain in performance. In this setting, on-policy algorithms struggle to exploit the limited reward information. The sample reuse in our GPI algorithms, on the other hand, allows sparse reward signals to be exploited across several policy updates while also leading to larger, more diverse batches of data at every update. Together, these benefits of sample reuse result in improved learning progress in difficult sparse reward settings.

VIII. CONCLUSION

In this work, we have developed a class of Generalized Policy Improvement algorithms that guarantee approximate policy improvement throughout training while reusing data from all recent policies. We demonstrated the theoretical benefits of principled sample reuse, and showed empirically that our generalized approach results in improved performance compared to popular on-policy algorithms across a variety of continuous control tasks from the DeepMind Control Suite. In particular, our algorithms accelerate learning in difficult sparse reward settings where on-policy algorithms perform poorly.

Because our methods use on-policy policy improvement algorithms as a starting point, our GPI algorithms only support the reuse of data from recent policies. An interesting avenue for future work includes the development of policy improvement guarantees that are compatible with the aggressive sample reuse in off-policy algorithms. Off-policy methods are often very data efficient when properly tuned, and the design of practical performance guarantees for these algorithms may increase their adoption in real-world control settings where large replay buffers are feasible.

[1]Code is available at `https://github.com/jqueeney/gpi`.

REFERENCES

[1] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annu. Rev. Control*, vol. 46, pp. 8–28, 2018.

[2] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, no. 1, pp. 253–279, 2019.

[3] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, 2016, pp. 1329–1338.

[4] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[5] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[6] A. Rajeswaran, I. Mordatch, and V. Kumar, "A game theoretic framework for model based reinforcement learning," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 7953–7963.

[7] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," arXiv:1911.11361, 2019.

[8] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020.

[9] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "MOReL: Model-based offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020.

[10] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 22–31.

[11] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2737–2752, 2019.

[12] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 176–188, 2022.

[13] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, no. 1, pp. 411–444, 2022.

[14] S. Paternain, M. Calvo-Fullana, L. F. O. Chamon, and A. Ribeiro, "Safe policies for reinforcement learning via primal-dual methods," *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1321–1336, 2023.

[15] L. Grossman and B. Plancher, "Just round: Quantized observation spaces enable memory efficient learning of dynamic locomotion," arXiv:2210.08065, 2022.

[16] J. Queeney, I. C. Paschalidis, and C. G. Cassandras, "Generalized proximal policy optimization with sample reuse," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021.

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

[18] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1889–1897.

[19] H. F. Song, A. Abdolmaleki, J. T. Springenberg, A. Clark, H. Soyer, J. W. Rae, S. Noury, A. Ahuja, S. Liu, D. Tirumala, N. Heess, D. Belov, M. Riedmiller, and M. M. Botvinick, "V-MPO: On-policy maximum a posteriori policy optimization for discrete and continuous control," in *Proc. 8th Int. Conf. Learn. Representations*, 2020.

[20] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Softw. Impacts*, vol. 6, p. 100022, 2020.

[21] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *Proc. 19th Int. Conf. Mach. Learn.*, 2002, pp. 267–274.

[22] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 3207–3214.

[23] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep RL: A case study on PPO and TRPO," in *Proc. 8th Int. Conf. Learn. Representations*, 2020.

[24] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters for on-policy deep actor-critic methods? A large-scale study," in *Proc. 9th Int. Conf. Learn. Representations*, 2021.

[25] J. Queeney, I. C. Paschalidis, and C. G. Cassandras, "Uncertainty-aware policy optimization: A robust, adaptive trust region approach," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 11, 2021, pp. 9377–9385.

[26] Y. Wang, H. He, X. Tan, and Y. Gan, "Trust region-guided proximal policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[27] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization," in *Proc. 35th Uncertainty Artif. Intell. Conf.*, vol. 115, 2020, pp. 113–122.

[28] Y. Cheng, L. Huang, and X. Wang, "Authentic boundary proximal policy optimization," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9428–9438, 2022.

[29] Q. Vuong, Y. Zhang, and K. Ross, "Supervised policy update for deep reinforcement learning," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[31] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1587–1596.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1861–1870.

[33] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a posteriori policy optimisation," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[34] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and Q-learning," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[35] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, "Q-Prop: Sample-efficient policy gradient with an off-policy critic," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[36] S. Gu, T. Lillicrap, R. E. Turner, Z. Ghahramani, B. Schölkopf, and S. Levine, "Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[37] W. Meng, Q. Zheng, Y. Shi, and G. Pan, "An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 5, pp. 2223–2235, 2022.

[38] R. Fakoor, P. Chaudhari, and A. J. Smola, "P3O: Policy-on policy-off policy optimization," in *Proc. 35th Uncertainty Artif. Intell. Conf.*, vol. 115, 2020, pp. 1017–1027.

[39] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[40] G. Novati and P. Koumoutsakos, "Remember and forget for experience replay," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 4851–4860.

[41] C. Wang, Y. Wu, Q. Vuong, and K. Ross, "Striving for simplicity and performance in off-policy DRL: Output normalization and non-uniform sampling," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 10 070–10 080.

[42] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[43] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Experience selection in deep reinforcement learning for control," *J. Mach. Learn. Res.*, vol. 19, no. 9, pp. 1–56, 2018.

[44] A. Kong, "A note on importance sampling using standardized weights," Tech. Rep. 348, Dept. Statist., Univ. Chicago, 1992.

[45] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee, "Diversity-driven exploration strategy for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[46] A. B. Tsybakov, *Introduction to Nonparametric Estimation*. Springer New York, NY, 2009.

[47] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[48] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1407–1416.