2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Dealing with Sparse Rewards in Continuous Control Robotics via Heavy-Tailed Policy Optimization

Souradip Chakraborty[1], Amrit Singh Bedi[1], Kasun Weerakoon[1], Prithvi Poddar[2], Alec Koppel[3], Pratap Tokekar[1], Dinesh Manocha[1]

Supplemental version including tech report, and video at `http://gamma.umd.edu/htspg/`

Abstract— **In this paper, we present a novel Heavy-Tailed Stochastic Policy Gradient (HT-PSG) algorithm to deal with the challenges of sparse rewards in continuous control problems. Sparse rewards are common in continuous control robotics tasks such as manipulation and navigation and make the learning problem hard due to the non-trivial estimation of value functions over the state space. This demands either reward shaping or expert demonstrations for the sparse reward environment. However, obtaining high-quality demonstrations is quite expensive and sometimes even impossible. We propose a heavy-tailed policy parametrization along with a modified momentum-based policy gradient tracking scheme (HT-SPG) to induce a stable exploratory behavior in the algorithm. The proposed algorithm does not require access to expert demonstrations. We test the performance of HT-SPG on various benchmark tasks of continuous control with sparse rewards such as 1D Mario, Pathological Mountain Car, Sparse Pendulum in OpenAI Gym, and Sparse MuJoCo environments (Hopper-v2, Half-Cheetah, Walker-2D). We show consistent performance improvement across all tasks in terms of high average cumulative reward without requiring access to expert demonstrations. We further demonstrate that a navigation policy trained using HT-SPG can be easily transferred into a Clearpath Husky robot to perform real-world navigation tasks.**

I. INTRODUCTION

Reinforcement learning (RL) has been employed with great success in several continuous control robotic tasks such as grasping [1], motion planning [2], and navigation [3], [4]. The key underlying idea in RL is to explore an unknown environment, collect rewards, and then move to maximize the reward collection. In practice, designing dense rewards is challenging for continuous control robotic tasks (see Fig.1) such as manipulation (joint angles of robots are continuous) and navigation (pose of robots and control inputs are continuous) [5]. Reward engineering for robotic tasks is difficult due to complex state space representations and usually requires manually-designed perception systems of the environment [6]. Hence, it's convenient to work directly with naturally specified sparse rewards [7]–[9]. For example, specifying a binary reward (1 for successful completion of a task and 0 otherwise) is significantly simpler than designing a dense reward structure. However, learning with sparse rewards is even more challenging since it results in the Hessian of the value function with respect to policy parameters being ill-conditioned [10].

[1]University of Maryland, College Park, MD, USA. E-mails: {schakra,amritbd,kasunw,pratap,dmanocha}@umd.edu.
[2]Indian Institutes of Science Education and Research. E-mail: {prithvid17}@iiserb.ac.in.
[3]JP Morgan AI Research, NewYork, NY, USA. E-mail: {alec.koppel}@jpmchase.com
This research was supported by Army Cooperative Agreement W911NF2120076.

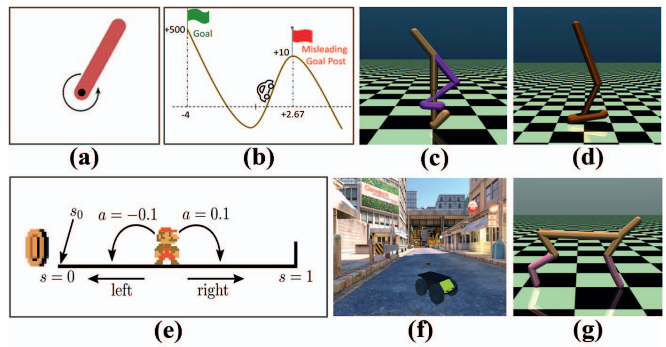


Fig. 1: Continuous control robotic environments used to train and evaluate our HT-PSG algorithm under sparse reward settings. (a) Sparse Inverted Pendulum; (b) Pathological Mountain Car; (c) Sparse Walker-2D; (d) One-legged hopper from Hopper-v2; (e) 1D Mario; (f) Clearpath Husky robot in Unity-based Simulation Environment; (g) Sparse cheetah. We observe that HT-SPG outperforms state-of-the-art algorithms in terms of cumulative reward returns and faster convergence.

In the literature, the issue of learning with sparse rewards is usually dealt either via reward shaping [11]–[13] or utilizing expert demonstrations [8], [14]–[17]. Also, authors in [18], [19] induce effective exploration via intrinsic curiosity or posterior variance, and [20], [21] utilizes information to motivate the exploration. Another line of work utilizes expert demonstrations to learn effectively in sparse reward environments [8], [9], [13], [22]. The main idea here is to either use available demonstrations to clone an expert's behavior (imitation learning) or just utilize demonstrations to provide additional rewards to guide the exploration [8], [9]. But the major limitation of these approaches is the dependence upon the quality of the expert demonstrations. If the demonstrations are sub-optimal or not good, these approaches fail badly. Apart from that, obtaining a high-quality demonstration is quite expensive, especially in robotic environments [1].

In contrast, we follow a different route and take motivation from the global convergence results in tabular MDP settings [23]. A crucial enabler for learning global optimal policies in [23] is the idea of *persistent exploration*, which helps to implicitly induce sufficient exploration in the state space. This ensures that the probability of taking any action in a given state is always non-zero, which would help to visit the complete state space and look for rewards. Recently, authors in [24], [25] have extended the idea of *persistent exploration* to continuous spaces and have proposed to utilize heavy-tailed policies to avoid convergence of policy gradient methods to spurious local maximas. Taking motivation from [24], we ask the following question

"*Can heavy-tailed policies make model-free RL sample efficient for practical robotics tasks that involve sparse reward structure without any expert demonstrations?*"

We answer this question in the affirmative in this paper and propose to utilize heavy-tailed policies (such as Cauchy) for policy parametrization along with a modified momentum-based policy gradient tracking to deal with the sparsity in reward. These heavy-tailed distributions appear heavily in fractal geometry [26], [27], finance [28], [29], pattern formation in nature [30], and networked systems [31], but has not been well investigated in RL framework with an exception of [32] where it is limited to applications such as simple outdoor navigation tasks. We summarize the main contributions of this paper as follows.

- We propose a novel algorithm to deal with sparse reward environments to train a policy in continuous state-action spaces. Our approach is fundamentally different from the introduction of heavy-tailed policy parametrization and avoids using expert demonstrations, which is a common practice in the existing literature. This provides a way to work with sparse reward environments without any reward shaping or demonstrations, which is very difficult otherwise. Additionally, our formulation is flexibly designed to efficiently incorporate prior demonstrations as well, if available.
- We observed that just replacing Gaussian policy parametrization with heavy-tailed (Cauchy) parametrization results in unstable behavior during the training. Hence, we propose a modified version of the momentum-based tracking [33] to control the variance of the stochastic gradient estimates.
- Finally, we show the efficacy of the proposed algorithm in both simulations and a real robot. The proposed algorithm demonstrates consistent performance improvement over a variety of benchmark problems (cf. Sec. V).

II. Related Work

Reward Shaping: Reward shaping is the most intuitive way to deal with sparse rewards. The idea was first appeared in [12] and further developed in recent works [13], [18], [20], [34]. The main idea revolves around intrinsic curiosity [18] and information gain based shaping [20]. Besides being simple, these methods come with the challenge of designing the additional reward functions which require expert supervision and demonstrations which are expensive. Additionally, it also induces expert-specific bias to the learning systems which ultimately leads the agent to explore only certain parts of the environment hindering the overall improvement.

Imitation Learning: Another line of work focuses on cloning an expert behavior called imitation learning (IL) [35]. Inverse reinforcement learning (IRL) is one way to do IL by extracting rewards from the given set of expert's trajectories for a given task [15], [16]. This issue of reward estimation was resolved by generative adversarial imitation learning (GAIL) algorithm by utilizing a discriminator to provide reward functions [36]. But the main drawback of IL-based approaches is that they do not utilize the feedback from the environment and behave according to the policies learned from demonstrations. Our approach in this work is fundamentally different, and we propose a method that works without demonstrations.

Learning from Demonstration: The idea here is to utilize expert's demonstrations to guide the standard learning procedure in RL algorithms [8], [9], [17], [37], [38]. Authors in [22], [39] proposed to include expert demonstrations to replay buffers and utilize them to accelerate the learning. The authors in [8] proposed an effective way to combine information from expert's policy to guide the exploration in the policy gradient algorithms. Mainly, the original reward function is modified to also include a term that accounts for the distance of current policy to the expert's policy. But as mentioned previously, the major drawback here is also their dependence upon the availability of demonstrations, which are hard to get in practice for continuous control problems. For instance, expert's demonstrations in [9] are obtained by running TRPO with dense rewards and then later used to train a policy with sparse rewards in the same environment. This could be difficult to achieve in practice. Therefore, we propose to modify the policy parametrization in continuous control environments to induce the required exploration in the learning procedure.

Heavy-Tailed Policy Parametrization: The idea of parametrizing policies via heavy-tailed distribution has appeared in the reinforcement learning literature [24], [40]. The authors in [40] proposed to utilize beta distribution for policy parametrization but are restricted to dense reward structure environments. Authors in [24] have focused on the development of heavy-tailed policy gradient to avoid convergence to local maxima and do not explicitly deal with sparse rewards. This work focus on sparse reward continuous control environments and extensive experimental evaluations to support the importance of heavy-tailed policy parametrization.

The paper is organized as follows. We start with the problem formulation in Sec. III, followed by proposed algorithm in Sec. IV. We present experimental results in Sec. V, and then conclude the paper in Sec. VI.

III. Markov Decision Problems with Sparse Rewards

When we formulate the continuous control robotics problems via reinforcement learning (RL), an autonomous robot interacts with the underlying environment by visiting different states in the state space $\mathcal{S}$. It starts from a particular state $s \in \mathcal{S}$, selects an action $a \in \mathcal{A}$ from the action space, and then transitions to another state $s' \in \mathcal{S}$ in the state space. The next state follows an unknown Markov transition density $\mathbb{P}(s'|s, a)$. After reaching state s', agent receives an instantaneous reward of $r(s, a)$ which quantifies the merit of decision a at state s. Mathematically, this frameworks is defined as Markov Decision Process (MDP) given by $\mathcal{M} := \{\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \gamma\}$, where $\gamma \in (0, 1)$ is the discount factor which decides the importance of future rewards for each instant. The state space $\mathcal{S} \subseteq \mathbb{R}^q$ and action space $\mathcal{A} \subseteq \mathbb{R}^p$ is continuous. Hence, we hypothesize that the agent selects actions $a_t \sim \pi(\cdot|s)$ over a time invariant parameterized distribution denoted by $\pi_{\boldsymbol{\theta}}(\cdot|s)$ for a given state s. The distribution $\pi(\cdot|s)$ is called a policy which controls the probability of taking a particular action a in given state s. The goal in the RL problem is to search for parameter $\boldsymbol{\theta}$ for the policy such that the average cumulative reward return (or value) is maximized given by :

$$V^{\pi_{\boldsymbol{\theta}}}(s)=\mathbb{E}\bigg[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\mid s_0=s, a_t\sim\pi_{\boldsymbol{\theta}}(\cdot|s_t)\bigg], \quad (1)$$

where $V^{\pi_{\boldsymbol{\theta}}}(s)$ is the value function with respect to state s, and s_0 denotes the initial state along a trajectory $\{s_t,a_t,r(s_t,a_t)\}_{u=0}^{\infty}$. Similar to fixing the initial state s_0, if we fix initial action as well $a_0=a$, the we can write the action-value function as $Q^{\pi_{\boldsymbol{\theta}}}(s,a)=\mathbb{E}\bigg[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\mid s_0=s, a_0=a, a_t\sim\pi_{\boldsymbol{\theta}}(\cdot|s_t)\bigg]$. We note that the expectation here is with respect to the product measure of policy $a_t\sim\pi(\cdot|s_t)$ and state transition density $s_{t+1}\sim\mathbb{P}(.|s_t,a_t)$. The selection of action a_t would control the possibility of visiting different state in the state space $\mathcal{S}$, and hence also responsible for exploring the state space. This also becomes important because in this work, we are specifically interested in environments where the rewards are sparse (cf. Sec. V). By sparse rewards we mean that they are available once in a while (see Fig. 1(e) 1D Mario) or there are high reward states available (see Fig. 1(b) pathological mountain car) but too far in the state space. Learning a good policy in such environments is a difficult task and that is the focus of this work. Hence, the goal here is to search over set of parameters $\boldsymbol{\theta}$ [41] given by

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}):=V^{\pi_{\boldsymbol{\theta}}}(s_0), \quad (2)$$

with $s_0\sim\rho_0(s)$. We note that the problem in (2) is non-convex with respect to optimization variable $\boldsymbol{\theta}$. Next, we mention the standard policy gradient algorithm to solve the problem in (2) and discuss challenges in the sparse reward settings.

Policy Gradient Algorithm. The key result which enables us to write policy gradient for the complicated objective in (2) is the Policy Gradient Theorem [41], which states that the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ can be written as

$$\nabla J(\theta)=\frac{1}{1-\gamma}\cdot\mathbb{E}\big[\nabla\log\pi_\theta(a|s)\cdot Q^{\pi_\theta}(s,a)\big], \quad (3)$$

and the expectation in (3) is over $(s,a)\sim\rho_\theta(\cdot,\cdot)$ where $\rho_\theta(s,a)=\rho_{\pi_\theta}(s)\cdot\pi_\theta(a|s)$ now denotes a valid probability distribution function also called as *discounted state-action occupancy measure* over continuous state and action spaces. From the expressions of $\rho_\theta(s,a)$ note that the selection of policy class has a significant affect on the eventually occupancy measure induced. In tabular MDP settings, to make sure the convergence to global optimal, an assumption of *persistent exploration* is needed [23], which is satisfied by making sure that $\pi(a|s)>0$ for all s and a. We remark that satisfying such assumption automatically takes care of the fact that we explore almost all parts of the state space because the probability of reaching any other state s' is not zero because of $\pi(a|s)>0$. Therefore, in tabular MDP, things work well even in the sparse reward settings. In contrast, in continuous action spaces, imposing such assumption $\pi(a|s)>0$ on the policy distribution $\pi(a|s)>0$ would violate the integrable assumption of probability distributions, and hence is not a valid assumption. So the induced exploration in the state space is mostly controlled by the policy distribution class we choose for parametrization. The standard policy parametrization class which is widely used in the literature is Gaussian [8], [9], [42], [43], and is given by

$$\pi_{\boldsymbol{\theta}}(a|s)=\mathcal{N}(a|\varphi(s)^\top\boldsymbol{\theta},\sigma^2), \quad (4)$$

where $\boldsymbol{\theta}$ controls the mean of the Gaussian, $\varphi(s)$ denotes the states feature representation $\varphi:\mathcal{S}\to\mathbb{R}^d$ with $d\ll q$, and σ^2 is fixed variance. We can make σ as a parameter as well but we avoid that for the sake of simplicity. Now, specifically for sparse reward settings, one major issue with Gaussian parametrization is that it would restrict the model transition to a state s' which is farther from current state s due to action selection $a\sim\mathcal{N}(\varphi(s)^\top\boldsymbol{\theta},\sigma^2)$ close to mean value. This induces a limited exploration for the algorithm, and it fails to learn in sparse reward environments. To deal with this issue, different techniques such as information maximization [20] and learning from demonstrations [8] are proposed. But the main disadvantages of such techniques are that entropy regularization required the estimation of the density function of occupancy which is quite expensive, and prior demonstrations could be quite bad and lead to completely irrelevant policies. Hence, to deal with such issues, instead of proposing any augmentation to existing techniques to handle sparse rewards, we resort to a new approach and propose to utilize heavy-tailed distributions to parameterize the policy $\pi_{\boldsymbol{\theta}}$. We explain this idea in detail in the next section.

IV. Proposed Heavy-Tailed Stochastic Policy Gradient for Sparse Rewards

In this section, we present the main idea of this work and develop a stable heavy-tailed stochastic policy gradient descent algorithm to deal with sparse reward settings.

A. Heavy-Tailed Policy Parametrization

As a first step towards developing such an algorithm, we propose to parameterize the policy by a class of heavy-tailed distributions. An example of heavy such parametrization is Cauchy distribution which is given by

$$\pi_{\boldsymbol{\theta}}(a|s)=\frac{1}{\sigma\pi(1+((a-\varphi(s)^\top\boldsymbol{\theta})/\sigma)^2)}, \quad (5)$$

where σ is the fixed scaling parameter. Other heavy-tailed distributions include the Extreme value distribution, Weibull distribution, log-normal distribution, Student's t distribution, Generalized Gaussian distribution, etc. The Laplace distribution has also fatter tails than the Gaussian distribution. In the financial literature, such distributions have been associated with the phenomenon of "black swan" events [28], [29].

With the policy parametrization specified, next goal is to compute the policy gradient mentioned in (3). But the challenge is the transition model dynamics are assumed to me unknown so it is not possible to evaluate $\nabla J(\theta)$ in closed form. So we take stochastic approximation approach and evaluate the stochastic gradient estimate. To write that, consider a randomized horizon $T_k\sim\text{Geom}(1-\gamma^{1/2})$ with trajectory sample $\{(s_0,a_0)\cdots(s_{T_k},a_{T_k})\}=:\xi_k(\boldsymbol{\theta}_k)$, then stochastic gradient can be written as

$$\nabla J(\boldsymbol{\theta}_k,\xi_k(\boldsymbol{\theta}_k)) \quad (6)$$
$$=\sum_{t=0}^{T_k}\gamma^{t/2}r(s_t,a_t)\cdot\bigg(\sum_{\tau=0}^{t}\nabla\log\pi_{\boldsymbol{\theta}_k}(a_\tau|s_\tau)\bigg),$$

Algorithm 1 Heavy-Tailed Stochastic Policy Gradient (HT-SPG)

Initialize : Initial parameter $\boldsymbol{\theta}_0$, momentum parameter β, discount factor γ, step-size η, and gradient estimate $\boldsymbol{g}_0$=0
Repeat for $k = 1, \ldots$
Sample two trajectories $\xi_k(\boldsymbol{\theta}_k)$ and $\xi_k(\boldsymbol{\theta}_{k-1})$ of length $T_k \sim \text{Geom}(1-\gamma^{1/2})$ using policies $\pi_{\boldsymbol{\theta}_k}$ and $\pi_{\boldsymbol{\theta}_{k-1}}$, respectively
Estimate $\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k))$, $\nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1}))$ via (6) and (12), respectively
Estimate $\boldsymbol{g}_k$ via (10)
Update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \boldsymbol{g}_k$
$k \leftarrow k+1$
Until Convergence
Return: $\boldsymbol{\theta}_k$

where $\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k))$ denotes the unbiased estimator of gradient $\nabla J(\boldsymbol{\theta}_k)$ at $\boldsymbol{\theta}_k$ (see [24, Lemma 1] for proofs) and and $\xi_k(\boldsymbol{\theta}_k)$ denotes the randomness in the estimate at k. Note the variable horizon length of the trajectories in (6) which is important to obtain an unbiased estimator. Otherwise, a fixed horizon length estimators where $T_k = H$ for all k (as in [43], [44]), results in a bias-variance tradeoff for gradient estimate [45]. Further, note the summation over two indexes in (6) t corresponds to the rollout trajectory, and τ collects score function till t from the starting. With the stochastic gradient defined in (6), the heavy tailed stochastic policy gradient iterate is given by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)), \tag{7}$$

where $\eta > 0$ denotes the step size. We note that the stochastic gradient in (7) can be zero because of the sparse nature of the rewards. For instance, it is possible that agent do not see a non-zero reward in the collected trajectory $\xi_k(\boldsymbol{\theta}_k)$ at k. Therefore, from (7), $\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)) = \mathbf{0}$ and we will end up not update parameter theta, and will take actions from the same policy at $k+1$ as well. We need an inherent exploration here to avoid getting stuck at this point. Interestingly, a heavy tailed policy comes to rescue here serves the purpose of selecting actions far from mean and induce sufficient exploration into the algorithm behavior. But also exhibits a downside as well. The resulting algorithm tends to be unstable to to heavy tails and probability of taking extreme actions. We mitigate this issue by introducing a momentum based gradient tracking to the proposed algorithm which is the focus of next subsection.

B. Stable Heavy-Tailed Stochastic PG Algorithm

The direct replacement of Gaussian policy parametrization with heavy-tailed policy parametrization results in an unstable behavior for the algorithm because the stochastic gradient estimates exhibit high variations from one sample to another. To deal with this issue, we need to invoke the idea of introducing momentum to stochastic gradient (SG) updates which has been successfully used in other machine learning approaches [46]. Hence, we replace the update in (7) as follows

$$\boldsymbol{g}_k = (1-\beta)\boldsymbol{g}_{k-1} + \beta \nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)), \tag{8}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \boldsymbol{g}_k, \tag{9}$$

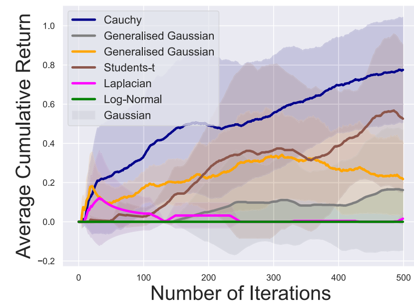


Fig. 2: In this figure, we show the importance of Cauchy as our heavy-tailed policy as compared to other policy parametrizations. We run tests on 1D Mario continuous control environment and plot the average reward return for different policy parametrizations. It is clear that Cauchy performs the best among all of them and achieves the highest reward return.

where β is the tuning parameter and update in (8) is called the momentum update. Note that for a small β (say $\beta = 0.2$) would results in utilizing the exponential average of past gradients rather than just considering the current stochastic gradient $\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k))$. This update is popular in the SG descent literature and achieves significant improvement empirically as compared to special case of $\beta = 1$ [13 from [33]] but does not result in theoretical gain. To address this issue, the authors in [33] have proposed a modified momentum based gradient tracking which result in provable variance reduction. With motivation from results in [33], we propose a novel gradient tracking scheme presented next for stochastic policy gradients with heavy-tailed policy parametrization as

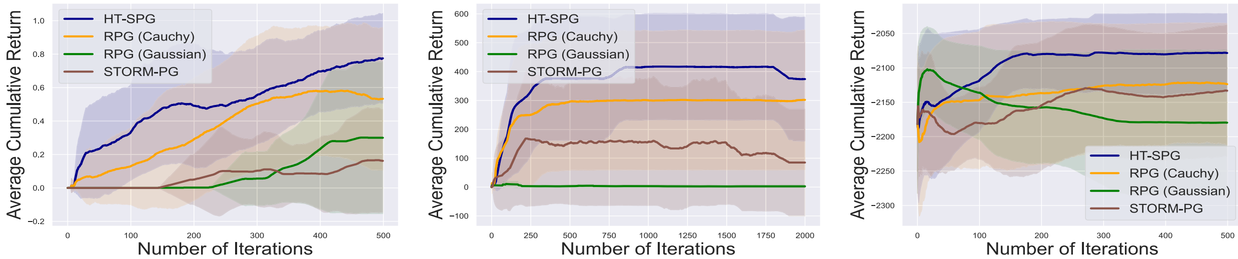
$$\begin{aligned}\boldsymbol{g}_k =& (1-\beta)\boldsymbol{g}_{k-1} + \beta \nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)) \quad (10)\\ &+ (1-\beta)(\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)) - \nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1}))),\\ \boldsymbol{\theta}_{k+1} =& \boldsymbol{\theta}_k + \eta \boldsymbol{g}_k, \quad (11)\end{aligned}$$

where $\nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1}))$ denotes the another stochastic gradient evaluated at instant k with policy parameter θ_{k-1}. The explicit expression is given by

$$\begin{aligned}&\nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1})) \quad (12)\\ &\quad = \sum_{t=0}^{T_k} \gamma^{t/2} r(s'_t, a'_t) \cdot \left(\sum_{\tau=0}^{t} \nabla \log \pi_{\boldsymbol{\theta}_{k-1}}(a'_\tau | s'_\tau) \right),\end{aligned}$$

where $\xi_k(\boldsymbol{\theta}_{k-1}) := \{s'_i, a'_i, r(s'_i, a'_i)\}_{i=0}^{T_k}$ denotes the trajectory generated bu using policy parameter $\boldsymbol{\theta}_{k-1}$ but at instance k. Note that there will be two Monte Carlo trajectories required to perform the update in (10). We remark an important difference of update in (10) to the gradient tracking proposed in [33]. The momentum step in [33, Eq. (2)] would require the use of $\nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_k))$ (to keep the stochastic quantity same) instead of $\nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1}))$ which we propose to use in this work. The use of term $\nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_k))$ has been proposed in the literature for reinforcement learning settings in [43] along with importance sampling weight adjustments to take care of the distributional shift which occurs due to the dependence of stochastic trajectory $\xi_k(\boldsymbol{\theta}_k)$ on $\boldsymbol{\theta}_k$. Next, we intuitively explain why it makes sense to use the update in (10) and it helps to reduce the variance of stochastic gradients, and hence results in a stable algorithm.

To understand it, let us consider the stochastic error introduced to the original gradient due to (10) as $\epsilon_k =$



(a) 1D Mario Environment [47]. (b) Pathological Mountain Car. (c) Sparse Pendulum.

Fig. 3: In this figure, we compare the performance of the proposed HT-SPG algorithm with RPG [42] and STORM-PG [43] which are state-of-the-art algorithms to solve the same problems without expert demonstrations. Here, RPG (Cauchy) denotes the RPG algorithm with Cauchy policy parametrization and we compared it to show that just replacing Gaussian with Cauchy is not the best thing to do. It works but results in a high variance in the reward returns as shown by the high confidence intervals of the yellow line. We plot the average cumulative reward return with respect to number of iterations/episodes for (a) 1D Mario environment [47], (b) Pathological Mountain Car (cf. 3(b)), and (c) Sparse Pendulum of OpenAI Gym environments. We note that the HT-SPG is able to achieve the highest reward return consistently in all the environments. Note: Total number of training samples = Batch size * No of Iterations.

$\boldsymbol{g}_k - \nabla J(\boldsymbol{\theta}_k)$. We note that ϵ_k defines the stochastic error in the gradient direction to perform the ascent update, and if we show that $\mathbb{E}\|\epsilon_k\|^2$ has a decreasing behavior with respect to k, this implies that the proposed momentum based update has resulted in variance reduction. Let us look at the explicit expression of ϵ_k as

$$\begin{aligned}\epsilon_k =&(1-\beta)\epsilon_{k-1} + \beta(\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)) - \nabla J(\boldsymbol{\theta}_k))\\ &+ (1-\beta)(\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)) - \nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1})))\\ &+ (1-\beta)(\nabla J(\boldsymbol{\theta}_k) - \nabla J(\boldsymbol{\theta}_{k-1})). \qquad (13)\end{aligned}$$

Next, note that it is the second, third, and fourth term on the right hand side of (13) which we need to control. We can easily control the second term on the right hand side of (13) by keeping β small. From the smoothness of J, we know that $\|\nabla J(\boldsymbol{\theta}_k) - \nabla J(\boldsymbol{\theta}_{k-1})\| \approx \mathcal{O}(\eta\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}\|)$ which can be controlled by step size η. The only remaining term is $\|\nabla J(\boldsymbol{\theta}_k, \xi_k(\boldsymbol{\theta}_k)) - \nabla J(\boldsymbol{\theta}_{k-1}, \xi_k(\boldsymbol{\theta}_{k-1}))\|$ which can also be assumed $\approx \mathcal{O}(\eta\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}\|)$ when $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_{k-1}$ are close to each other. This is possible because of the dependence of trajectories $\xi_k(\cdot)$ on $\boldsymbol{\theta}$ which is not the case in [33]. Therefore, by controlling β and η, it is possible to develop a stable algorithm with heavy-tailed policy parametrizations. We summarize the algorithm steps in Algorithm 1. Further, we extensively test the empirical performance of the purposed algorithm in different sparse environments in the next section and show the performance benefits achieved in practice. We defer the theoretical analysis of the proposed algorithm to the future scope of this work.

V. Experiments

In this section, we proceed to perform extensive experimental validation of the proposed algorithm. We start our analysis with low dimensional sparse environments such as 1D Mario [47], Pathological Mountain Car (cf. Fig. 1), and Sparse Pendulum [50]. Secondly, we incorporate Sparse MuJoCo environments namely Hopper-v2 as done in [9] to highlight our method's performance. Next, we utilize a high-fidelity unity simulator to train a navigation policy with HT-SPG for a differential drive robot model. Finally, we demonstrate that the navigation policy trained using our method can be transferred to a real robot without significant performance degradation.

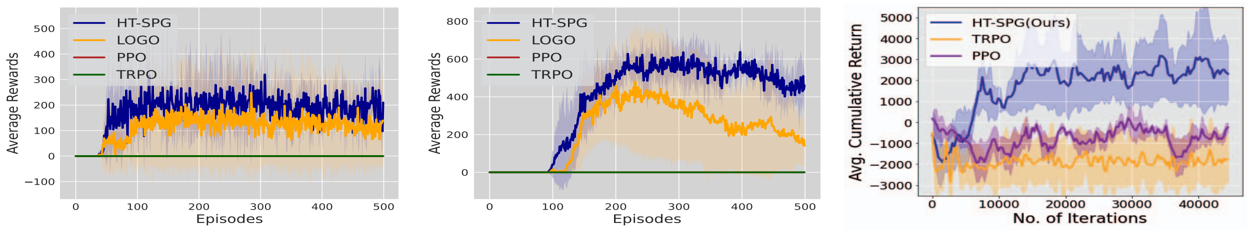
Importance of Policy Parametrization: Before discussing the main experimental results, we start by demonstrating (see Fig. 2) the limitations of light-tail policy parametrization and emphasize the importance of using heavy-tail distributions such as Cauchy for policy parametrization. Fig. 2 shows the average cumulative reward return for different policy parametrizations in a 1D Mario environment. We demonstrate that Cauchy distribution-based policy is able to achieve the highest reward return in the most sample-efficient manner. This is mainly due to the better exploratory behavior achieved by the Cauchy-based policy as compared to other policies. Hence, we will be using Cauchy policy parametrizations for the rest of the experiments. We detail the different environment settings as follows.

A. Learning Without Demonstrations

In this subsection, we run experiments in sparse reward environments and compare them against other state-of-the-art algorithms which operate without any access to expert demonstrations. The details of environments are provided in Appendix A in the supplementary [51]. We run the proposed algorithm HT-SPG in the above-mentioned environments and compare it with other state-of-the-art existing algorithms with light-tailed policy parametrization (Gaussian) such as RPG [42], and STORM-PG [43]. There is a state-of-the-art algorithm to solve continuous control problems without any demonstrations. We present the results in Fig. 4, where RPG (Cauchy) denotes RPG algorithm with Cauchy policy parametrization. It is included to show that just replacing Gaussian (RPG (Gaussian)) with Cauchy is not sufficient to achieve the desired performance, and it results in unstable behavior which exhibits high variance in the reward returns. This issue is corrected by using momentum-based tracking in HT-SPG. In all these classic continuous control environments with sparse rewards, our proposed HT-SPG algorithm outperforms all the other methods based on light-tail distribution, emphasizing the significance of heavy-tailed parameterization in learning under complex and sparse scenarios. We also remark that HT-SPG is extremely easy to implement and train and can be integrated with any learning task endowed with complex and sparse rewards distribution for enhanced performance.

B. MuJuCo Sparse Environments

In this section, we consider the complex sparse MuJoCo environments of Hopper (see Fig. 1(d)) and test the performance of the proposed HT-SPG algorithm. We compare it



(a) Sparse Half-Cheetah [47].

(b) Sparse Hopper-v2

(c) Husky Robot Navigation.

Fig. 4: We compare the the proposed HT-SPG algorithm with SOTA LOGO [9] PPO and TRPO [48], [49]. HT-SPG outperforms the existing algorithms and achieves higher rewards in the given settings. The total number of training samples = Batch size * No of Iterations. Fig. 4(c) shows the comparison between HT-SPG based sparse navigation policy vs PPO and TRPO policies. Our method obtains the highest average reward returns with faster policy convergence.

with the state-of-the-art LOGO algorithm [9]. The state and action spaces for these environments are no longer scalar and require us to deal with multivariate distributions for the policy parametrizations. State-space is 12-dimensional, action space is 3-dimensional linear reward for forward progress, and a quadratic penalty on a joint effort to produce the reward with a bonus of $+1$ for being in a non-terminal state. The sparsity in reward structure is obtained by reducing the events at which reward feedback is provided. Specifically, we provide a reward of $+1$ only after the agent moves forward over 2 units from its initial position. We present the performance of HT-SPG as compared to the LOGO algorithm, PPO, and TRPO in Fig. 1(d). Since the performance of LOGO was optimized to operate with demonstrations, we considered the same learning environment with minimal demonstrations for the proposed HT-SPG algorithm as well. We note that the proposed algorithm is able to outperform LOGO, PPO, and TRPO by a good margin and exhibit better sample efficiency. The results for sparse walker-2D are available in supplementary material [51].

C. Real-World Experiments

To show the efficacy of the proposed approach in real-world scenarios, we consider a collision-free robot navigation task using a differential drive robot (a Clearpath Husky model) in a high-fidelity Unity simulator. The robot model is equipped with a VLP16 Velodyne LiDAR. In this experiment, we have a two dimensional continuous action $a = (v, \omega)$ (i.e. linear and angular velocities) and a state $s = [d_{goal}, \alpha_{goal}, l_{obs}]$ that includes goal reaching and obstacle avoidance related observations. Here, $d_{goal} \in \mathbb{R}^+$ and $\alpha_{goal} \in [0, \pi]$ are distance and angle to the goal w.r.t. the robot's current position respectively. And l_{obs} is a vector that contains distances to the obstacles around the robot obtained from a 2D LiDAR scan. We only use 2D scans primarily due to the low dimensionality compared to the 3D point cloud.

Further, we specifically define a set of sparse rewards r_h, r_d, and r_{obs} to train a policy using HT-SPG for collision-free robot navigation in the simulation environment. Let r_h and r_d be the rewards to encourage the robot to reach the goal, defined as $r_h = \mathbb{1}_{\{|\alpha_{goal}| \leq \pi/6\}}$, and $r_d = \beta_1 \mathbb{1}_{\{\frac{R_g}{2} - \epsilon \leq d_{goal} \leq \frac{R_g}{2} + \epsilon\}} + \beta_2 \mathbb{1}_{\{-\epsilon \leq d_{goal} \leq \epsilon\}}$, where R_g is the straight line distance to the goal from the robot's starting position, $\epsilon \in [0, 1], \beta_1$ and β_2 are adjustable scalar parameters. Intuitively, r_d is non-zero only when the robot reaches half a distance to the goal or the actual goal location. Similarly, r_h is non-zero only when the robot maintains a heading angle in the $[-\pi/6, \pi/6]$ range. Further, we define the obstacle avoidance penalty r_{obs} as,

Fig. 5: Trajectories when navigating a Clearpath Husky robot in both simulated and real Urban environments using a policy trained with HT-SPG under sparse reward settings. The trained navigation policy can perform equally well in real world environments.

$$r_{obs} = \begin{cases} -50 & \text{if } \min(l_{obs}) \leq l_{collision}, \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

where $l_{collision}$ is a safety distance threshold that the robot should maintain with obstacles. The final sparse reward $r = r_h + r_d + r_{obs}$ is used to train the navigation policy with HT-SPG and other algorithms PPO and TRPO for comparisons. These comparisons are presented in Fig. 4(c). We observe that HT-SPG receives the highest reward returns compared to PPO and TRPO. We evaluate the navigation performance of our trained policy in both simulated and real-world environments. We demonstrate that our trained policy can be transferred to a real Husky robot without significant performance degradation. The real robot is equipped with a VLP16 Velodyne LiDAR, a laptop with an Intel i9 CPU, and an Nvidia RTX 2080 GPU. Trajectories generated during simulated and real outdoor experiments are presented in Fig. 5.

VI. Conclusion, Limitations, and Future Work

In this work, we proposed a novel approach to deal with sparse reward in continuous control robotic tasks. Instead of relying on reward shaping or seeking information from expert demonstrations, we utilize heavy-tailed policy parametrizations along with momentum-based gradient tracking to learn in sparse robotics environments. We prove the efficacy of the proposed ideas on various robotics tasks of OpenAI Gym and MuJoCo environments. We also provided a real-world demonstration of the proposed algorithm. The main limitation of the current approach is that we cannot prove any theoretical convergence guarantees for the proposed approach, an open problem for future work.

References

[1] O. Kilinc and G. Montana, "Reinforcement learning for robotic manipulation using simulated locomotion demonstrations," *Machine Learning*, pp. 1–22, 2021.

[2] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[3] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[4] K. Weerakoon, A. J. Sathyamoorthy, U. Patel, and D. Manocha, "Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9447–9453.

[5] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[6] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv preprint arXiv:1904.07854*, 2019.

[7] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. Aparicio Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5548–5555.

[8] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *International conference on machine learning*. PMLR, 2018, pp. 2469–2478.

[9] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," *arXiv preprint arXiv:2202.04628*, 2022.

[10] P. Rauber, A. Ummadisingu, F. Mutz, and J. Schmidhuber, "Reinforcement learning in sparse-reward environments with hindsight policy gradients," *Neural Computation*, vol. 33, no. 6, pp. 1498–1553, 2021.

[11] N. Botteghi, B. Sirmacek, K. A. Mustafa, M. Poel, and S. Stramigioli, "On reward shaping for mobile robot navigation: A reinforcement learning and slam based approach," *arXiv preprint arXiv:2002.04109*, 2020.

[12] M. J. Mataric, "Reward functions for accelerated learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 181–189.

[13] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[14] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[15] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[16] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[17] G. Libardi, Gabriele anfd De Fabritiis and S. Dittert, "Guided exploration with proximal policy optimization using a single demonstration," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6611–6620.

[18] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[19] S. Chakraborty, A. S. Bedi, A. Koppel, B. M. Sadler, F. Huang, P. Tokekar, and D. Manocha, "Posterior coreset construction with kernelized stein discrepancy for model-based reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2206.01162

[20] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," *Advances in neural information processing systems*, vol. 29, 2016.

[21] S. Chakraborty, A. S. Bedi, A. Koppel, M. Wang, F. Huang, and D. Manocha, "Steering: Stein information directed exploration for model-based reinforcement learning," 2023. [Online]. Available: https://arxiv.org/abs/2301.12038

[22] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[23] J. Mei, C. Xiao, C. Szepesvari, and D. Schuurmans, "On the global convergence rates of softmax policy gradient methods," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6820–6829.

[24] A. S. Bedi, A. Parayil, J. Zhang, M. Wang, and A. Koppel, "On the sample complexity and metastability of heavy-tailed policy search in continuous control," *arXiv preprint arXiv:2106.08414*, 2021.

[25] A. S. Bedi, S. Chakraborty, A. Parayil, B. M. Sadler, P. Tokekar, and A. Koppel, "On the hidden biases of policy mirror ascent in continuous action spaces," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 1716–1731. [Online]. Available: https://proceedings.mlr.press/v162/bedi22a.html

[26] J. E. Hutchinson, "Fractals and self similarity," *Indiana University Mathematics Journal*, vol. 30, no. 5, pp. 713–747, 1981.

[27] B. B. Mandelbrot, *The fractal geometry of nature*. WH freeman New York, 1982, vol. 1.

[28] N. N. Taleb, *The black swan: The impact of the highly improbable*. Random house, 2007, vol. 2.

[29] J. B. Taylor and J. C. Williams, "A black swan in the money market," *American Economic Journal: Macroeconomics*, vol. 1, no. 1, pp. 58–83, 2009.

[30] D. Avnir, O. Biham, D. Lidar, and O. Malcai, "Is the geometry of nature fractal?" *Science*, vol. 279, no. 5347, pp. 39–40, 1998.

[31] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.

[32] K. Weerakoon, S. Chakraborty, N. Karapetyan, A. J. Sathyamoorthy, A. Bedi, and D. Manocha, "Htron: Efficient outdoor navigation with sparse rewards via heavy tailed adaptive reinforce algorithm," in *6th Annual Conference on Robot Learning*.

[33] A. Cutkosky and F. Orabona, "Momentum-based variance reduction in non-convex sgd," *Advances in neural information processing systems*, vol. 32, 2019.

[34] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," *arXiv preprint arXiv:1706.01905*, 2017.

[35] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[36] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[37] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[38] J. Chen and W. Xu, "Policy gradient from demonstration and curiosity," *arXiv preprint arXiv:2004.10430*, 2020.

[39] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[40] P.-W. Chou, "The beta policy for continuous control reinforcement learning," Ph.D. dissertation, Master's thesis. Pittsburgh: Carnegie Mellon University, 2017.

[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[42] K. Zhang, A. Koppel, H. Zhu, and T. Basar, "Global convergence of policy gradient methods to (almost) locally optimal policies," *SIAM Journal on Control and Optimization*, vol. 58, no. 6, pp. 3586–3612, 2020.

[43] H. Yuan, X. Lian, J. Liu, and Y. Zhou, "Stochastic recursive momentum for policy gradient methods," *arXiv preprint arXiv:2003.04302*, 2020.

[44] M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli, "Stochastic variance-reduced policy gradient," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4026–4035.

[45] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] G. Matheron, N. Perrin, and O. Sigaud, "The problem with ddpg: understanding failures in deterministic environments with sparse rewards," *arXiv preprint arXiv:1911.11679*, 2019.

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

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

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

[51] "Technical report for "dealing with sparse rewards in continuous control robotics via heavy-tailed policy optimization"," 2022. [Online]. Available: https://tinyurl.com/43v3783j