iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.45509

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

Tracking Locomotion using Reinforcement Learning

Rutuja Jadhav[1], Prof. Priya D[2], Dr. Mamatha G S[3]
[1]*Student,* [2]*Assistant Professor,* [3]*Associate Dean, Information Science and Engineering, RV College of Engineering, Bangalore, India*

Abstract: ***This article presents the concept of reinforcement learning, which prepares a static direct approach for consistent control problems, and adjusts cutting-edge techniques for testing effectiveness in benchmark Mujoco locomotion tasks. This model was designed and developed to use the Mujoco Engine to track the movement of robotic structures and eliminate problems with assessment calculations using perceptron's and random search algorithms. Here, the machine learning model is trained to make a series of decisions.***
The humanoid model is considered to be one of the most difficult and ongoing problems to solve by applying state-of-the-art RL technology. The field of machine learning has a great influence on the training model of the RL environment. Here we use random seed values to provide continuous input to achieve optimized results.
The goal of this project is to use the Mujoco engine in a specific context to automatically determine the ideal behavior of the robot in an augmented reality environment. Enhanced random search was introduced to train linear guidelines for achieving the efficiency of Mujoco roaming tasks. The results of these models highlight the variability of the Mujoco benchmark task and lead to efficiently optimized rewards
Keywords: Reinforcement Learning, Policies, Benchmark, Mujoco.

I. INTRODUCTION

Reinforcement learning provides a solution for continuous input dynamic models. The location description is that the AI model is ready to create a collection of decisions.
The head knows how to pack the lens into a defective, perhaps complex condition. When composing, you encounter a playful state of artificial thinking ability. Artificial science skills are rewarded or trained for the activities that item engineers perform to get their machines to do what they need. This paper uses a model-free reinforcement learning method to train agents' linear policies to achieve optimized rewards based on advanced random search algorithms [1]. To implement this task, a trained policy is executed to create an augmented reality environment where the agent chooses the best action based on the current state and gets the most reward.
The Augmented reality is one of the best technique which can be used in real world environment that will help to enhance the virtual visual components by combining all the elements for a person's perception of real world. We use augmented reality to improve the way users visualize the robot's learning process.
This is because the user understands the AI training process and the robot's perception is the environment [2]. It is very clear that the process of manually designing a robot is a very rigorous effort for an engineer. Model-Free Reinforcement Learning (RL) aims to provide an off-the-shelf solution for controlling dynamic systems without the need for a model of system dynamics. Such methods have succeeded in creating RL agents that outperform human players in games such as video games and Go [20, 33]. While these results are impressive, model-free methods have not yet been successfully applied to control physical systems other than research demos. Unfortunately, the current trends in RL research conflict these obstacles [3]. Searching for sample-efficient methods (that is, methods that require little data) generally requires the development of increasingly complex methods. This increased complexity poses a reproducibility crisis [4]. Recent studies have shown that many RL methods are not robust to hyper parameter changes, random starting values, or different implementations of the same algorithm [12, 13].
In our work, agents use various linear policies as input to track their behavior in the environment [5]. The most popular continuous control benchmark is the Mujoco motion task [6], and the humanoid model is considered "one of the most difficult continuous control problems that can be solved with state-of-the-art RL technology [7] [8]. Reinforcement learning provides a solution for continuous control dynamic models.

II. RELATED WORK

1) Vassilios Tsounis, Mitija Alge, Joonha Lee et al [1], proposed new technique for training neural network policies for terrain-aware movement, combining model-based motion planning with state-of-the-art methods for reinforcement learning. The author used an approach known as the Markov decision process to evaluate dynamic feasibility criteria. To train the strategy, we used the strategic gradient method to perform kick-based movements. The authors also proposed a new method for model-free reinforcement learning. H. The Terrain Recognition Planner generates a set of actions that the robot uses to move towards the target. The other is a foot motion controller to correct and deal with disturbances.
2) Ziyi Zang, Samuel Micah Akai-Nettey et al [2], proposed a treatise that presents reinforcement learning through an augmented reality platform. In this task, we will introduce an augmented reality interface and a Lego® Spike robot to introduce students to the concept of RL. The students were presented with the concept of RL. 1) What are the statuses, actions and rewards? 2) Exploration and exploitation. 3) Q table for robots to make decisions. This task aims to solve real benchmarking problems, train the robot in a simulated environment, and track the robot's movements.
3) Octavio Villarreal, Victor Barasuol et al [3] Proposed a new control strategy for dynamic leg movement. This strategy is based on two main elements. One is a contact sequence task that provides a secure foundation based on a convolutional neural network and seeks a secure foundation to perform a rapid and continuous assessment of the terrain. The power of this strategy is harnessed through realistic on-board sensors and simulations of the hydraulically actuated four-legged robot HyQ Real that traverses rugged terrain under computational conditions.
4) Angelo Bratta, Romeo Orsolino, Michele Focchi , Victor Barasoul et al [4] The proposed leg-locomotion nonlinear trajectory optimization based on simplified dynamics helps to achieve robust locomotion in difficult terrain constraints and focuses on incorporating joint torque limiting constraints, these limitations. Approximate how it is converted to permissible contact force as it is mapped to the leg. The experimental contributions of this paper consist of providing optimal orbital hardware from the formulation.
5) Mnih et al. [5] It has been proposed that the policy of Atari video games and MuJoCo models can be quickly trained by asynchronously parallelizing the actor-critical methods common to decentralizing the policy gradient algorithm. Formerly Schulman et al. [30] introduced a generalized benefit estimation (GAE) method for estimating benefits. This can reduce the variance with less bias than the previous method.
6) Haarnoja et al. [6] The high variance of the proposed gradient estimation is not the only hurdle that the gradient method must overcome. The optimization problems that occur in RL are very non-convex and there are many ways to find the suboptimal local optimization. To deal with this problem, Haarnoja etc. [6] proposes a soft Q-learning algorithm for learning multimodal stochastic policies through entropy maximization, leading to better exploration in environments with multimodal reward landscapes. Recently, Haarnoja etc. [6] We combined this idea with the Actor Criticism Framework to form the Soft Actor Criticism (SAC) algorithm. This is an out-of-policy actor critique method that aims at both the rewards and entropy that actors expect to maximize their stochastic policies. From another direction simplified the search space using linear guidelines. They used a natural gradient, which is a policy gradient that fits the policy's parameter space metrics, to train the linear policy of the MuJoCo locomotion task.
7) Alexander L Mitchell, Martin Engelcke et al [7]. The proposed complex dynamic and terminal constraints are represented by high-level semantic indicators and deeply captured statistical representations of possible joint configurations while represented by learned classifiers operating in latent space. Generative model. The use of MPC provided a systematic and robust way to address the problem of quadrupedal walking. Nevertheless, it generally does not take into account future terrain within the forecast. Instead, it reacts to the terrain and relies on continuous updates of the state to provide sufficient robustness to initialize the optimization. Using terrain information to improve the performance of MPC strategies remains a challenge.

III. REINFORCEMENT LEARNING

Reinforcement learning is a machine learning training method based on rewarding desirable behaviors and / or punishing unwanted behaviors. In general, reinforcement learning agents can detect, interpret, perform actions, and learn through trial and error within the environment [9][10]. The world of agents is usually represented as a Markov decision process (MDP), 5 tuples ‒Where S is the set of states, A is the set of actions, T: $S \times A \rightarrow \Pi (S)$ is the transition function that maps the action and the probability of moving to a new state given the current state, R. : $S \times A \rightarrow$ R-rewards the action in a given state, $\gamma \in [0, 1)$. Discount factor. Consider an episode task that starts in the initial state s0 and starts a new episode when the agent reaches the stem in the final state [11]. At each interaction step, the agent observes its state and selects an action according to policy π: $S \rightarrow A$.

The agent's goal is to learn the optimal strategy π ∗ to maximize the long-term expected total of discounted rewards [12][13]. One way to learn the optimal strategy is to learn the optimal behavioral value function Q ∗ (s, a). This shows the expected total discount reward for performing action a in state s and then executing the next strategy π :

Q ∗（s、a）= R（s、a）+γXs 0 T（s 0 | s、a）×maxa0Q ∗（s 0、a0）

The goal of the RL algorithm is to almost solve problem (1) by making as few calls to the environment as possible. The number of environmental queries required to solve problem (1) is called the complexity of the environment or the complexity of the sample [14]. The best trained controller should be able to capture the key features of motion in a low-dimensional model and transform it into a stable walking motion that works on the hardware [15][16][17].

Some important terms that describe the basic elements of the RL problem are:

1) *Environment:* The physical world in which the agent operates
2) *Status:* Current status of the agent
3) *Reward:* Feedback from the environment
4) *Policy:* How to map the state of an agent to an action.
5) *Value:* Future rewards that agents will receive if they perform an action in a particular state.

IV. PROBLEM FORMULATION

To Design and develop the Reinforcement Learning model to train and track the locomotion of a RL agent using Mujoco engine to eliminate the issues faced in normal philosophy assessment calculations using random search algorithm. Such problems can be abstractly formulated as max

θ∈Rd Eξ [r(πθ, ξ)] ,

V. AUGMENTED RANDOM SEARCH

Note that the problem formulation (1) aims to optimize reward by directly optimizing over the policy parameters θ. We consider methods which explore in the parameter space rather than the action space. This choice renders RL training equivalent to derivative-free optimization with noisy function evaluations. One of the simplest and oldest optimization methods for derivative-free optimization is Augmented random search [18][19]. We are currently introducing augmented random search to extend the effective heuristics used in deep reinforcement learning. The main form of our technique is derived from basic random search. Augmented Random Search is a model free reinforcement learning a modified version of Basic Random Search Algorithm [20]. The basic idea behind reinforcement learning algorithms is to develop an algorithm that finds the best policy that can maximize agent rewards while applying that particular policy within the environment during an episode [21]. When maximizing the reward, it means that we are actually optimizing the process-we are trying to find the best value that gives the best result or the best objective function. In the ARS setup, agents learn from the environment, receive rewards based on their performance in the environment, and adjust weighted inputs in relation to the rewards received. As a result, the agent avoids repeating previous mistakes, but performs subsequent actions along with those lucrative rewarding actions, with the help of Perceptron[22]. The environment rewards each agent movement / action, while Perceptron chooses a total reward so that the weighted input can be adjusted. Perceptron-Takes input from the environment, applies some weights to it, sums it up, divides it by the standard deviation, and returns the output to the environment.

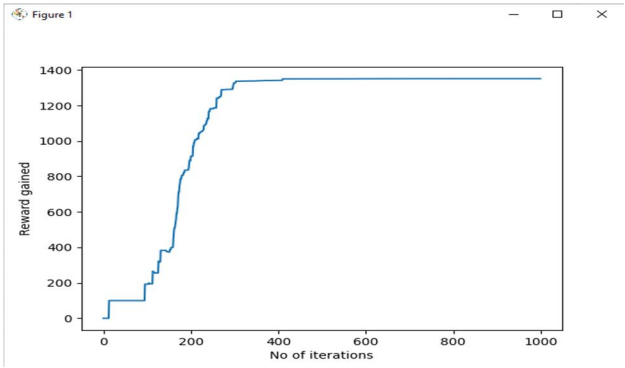


Fig 1. Graph of No of Iterations Vs Rewards Gained

Fig 1 describes the standard deviation of all the prizes at every step while preparing HalfCheetah-v1.

The Q-Learning method is used to learn the policies of a action taken for each particular state plotted in the above graph. The goal of the Q-Learning algorithm is to iteratively train the optimal Q values obtained after each action in RL environment and form a Q-Matrix to help agents at each step to take best actions to obtain positive feedback.

VI. SYSTEM ARCHITECTURE

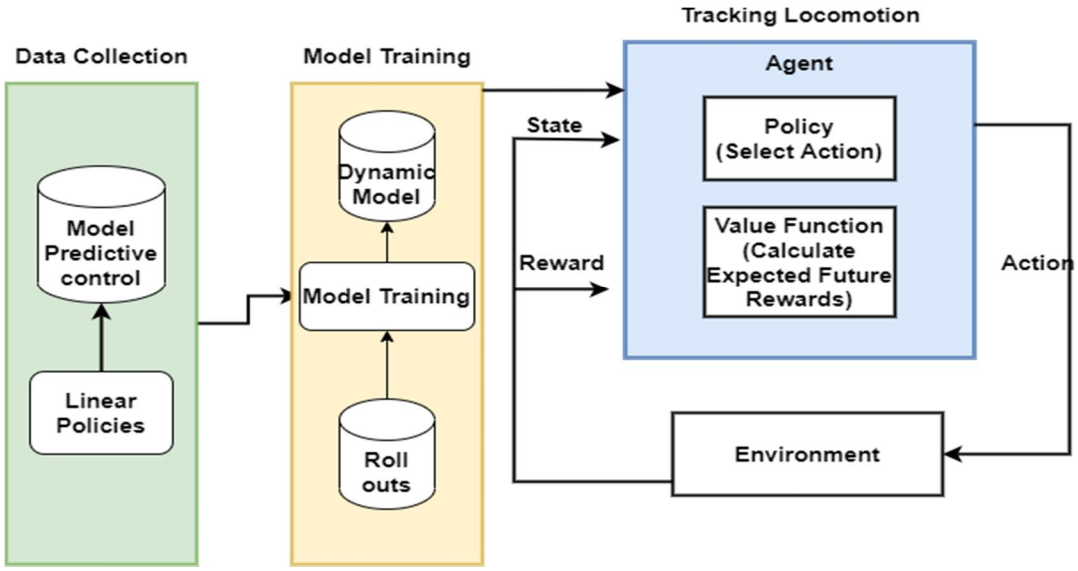


Fig 2. System Architecture

The fig 2 Represents the Tracking Locomotion system architecture using the RL project. The system architecture here describes the project flow and the components of the project. First, the project uses linear policies as input , here we are using policy based approach to obtain rewards i.e This approach finds the best policy for highest reward in future without using the value function and these policies are applied q-learning algorithm to learn the policies which forms a Q-matrix where the values are updated at each step for the action performed for a particular situation to get the efficiency path. The policy is trained with ARS and the agent takes action to predict the optimized reward by learning from its experience using trial and error method.For each good behavior or action the agent receives a positive feedback and for each bad behavior the agent will receive a negative feedback. So basically the agent is automatically learning from its experience and the feedbacks received from environment without the labelled data.

The agent interacts with environment and explores itself. The goal of this agent to improve its efficiency by learning from its mistake with the feedbacks received without human intervention [23].

VII. STANDARD DEVIATION ΣR

As the training of policies progresses, random search in the parameter space of policies can lead to large variations in the rewards observed across iterations[24][25]. As a result, it is difficult to choose a fixed step-size α which does not allow harmful changes between large and small steps.

To address the large variations of the differences $r(\pi M+\nu\delta) - r(\pi M-\nu\delta)$, we scale the update steps by the standard deviation σR of the 2N rewards collected at each iteration[26]. To understand the effect of scaling by σR, we plot standard deviations σR obtained during training a policy for the HalfCheetah-v1 model in Figure 1. The standard deviations σR have an increasing trend as training progresses.

This behavior occurs because perturbations of the policy weights at high rewards can cause HalfCheetah-v1 to fall early, yielding large variations in the rewards collected. Therefore, without scaling by σR, our method at iteration 300 would be taking steps which are a thousand times larger than in the beginning of training[27]. The same effect of scaling by σR could probably be obtained by tuning a step-size schedule. However, our goal was to minimize the amount of tuning required, and thus we opted for the scaling by the standard deviation[28][29].

VIII. EXPERIMENTAL RESULTS

A. Implementation Details

We implemented a parallel version of Algorithm using the Python library Ray. To avoid the computational bottleneck we used the Python Library Ray to perform the equivalent adaptation of the algorithm. Maintain a strategic distance to the computational barrier δ of stimulus transmission. Our code sets the random seeds for the random generators of all the workers and for all copies of the OpenAI Gym environments held by the workers [30]. All these random seeds are distinct and are a function of a single integer to which we refer as the random seed. Furthermore, we made sure that the states and rewards produced during the evaluation rollouts were not used in any form during training. Open AI Gym offers a unique Mujoco exercise benchmark reward capacity [31]. We used these reward features to evaluate the display of direct orders generated by the default enhanced random search.

B. Results on the MuJoCo locomotion Tasks

We evaluate the performance of ARS on MuJoCo roaming tasks included in OpenAI Gym v0.9.3. OpenAI Gym provides benchmark reward features for various MuJoCo locomotion tasks [32]. We used these standard reward functions to evaluate the performance of ARS-trained linear policies. The reported premiums received by the policy were the average of over 100 independent rollouts.

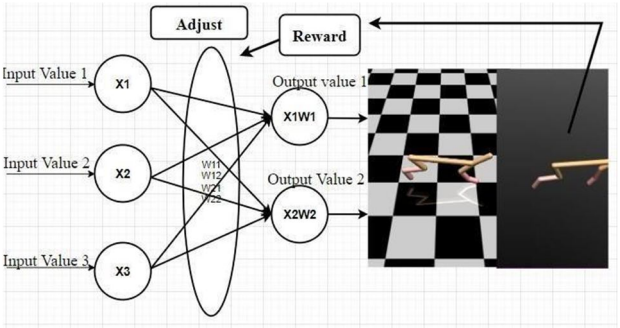


Fig 3. Feedback Mechanism of HalfCheetah-v1

Fig 3 describes that for a thorough evaluation, 100 different random seeds from intervals [0, 10000] were sampled evenly and randomly. ARS trains guidelines for all MuJoCo roaming tasks. In addition, ARS can train policies most of the time while using a competitive number of episodes.

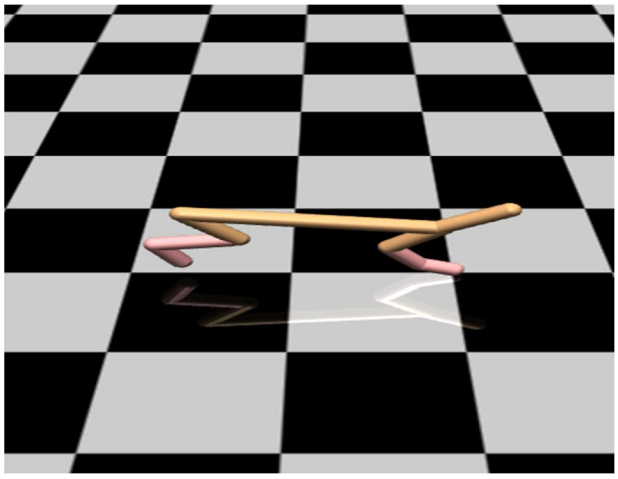


Fig 4 Simulation Results of HalfCheetah-v1

Fig 4 shows the simulation behavior of the HalfCheetah-v1 agent. The value approximation-based method does not work because the half-cheetah model uses a continuous action space instead of a discrete action space. Also Value functions usually need to use argmax to determine the best policy [33]. This makes working with continuous action spaces very unrealistic because it requires discretization of the action space. Therefore, we use the policy gradient method as the algorithm of choice.

IX. CONCLUSION

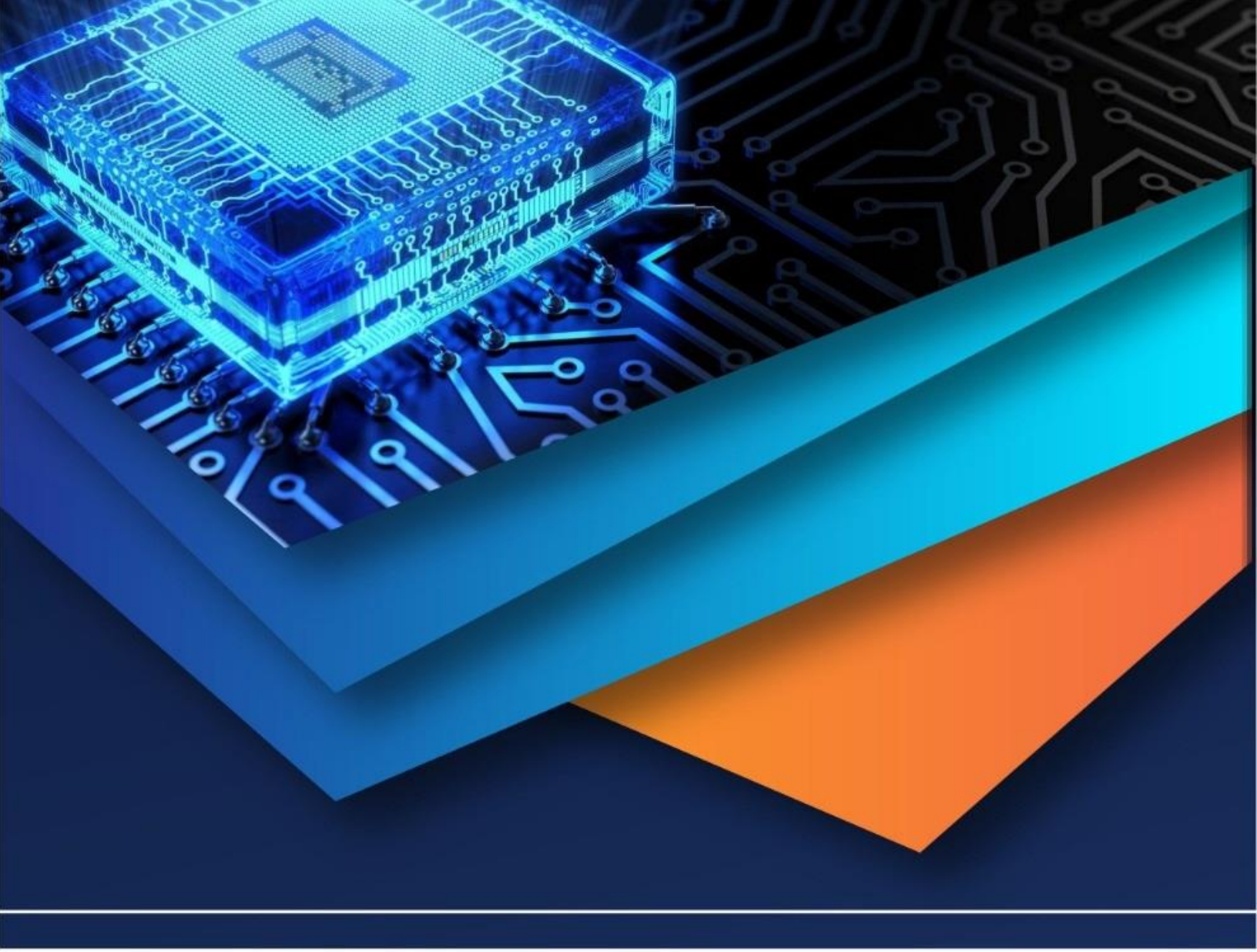
We attempted to find the simplest model-free RL algorithm that works well with the continuous control benchmarks used in the RL literature [34]. Several algorithm extensions have shown that advanced random searches can be used to train linear guidelines to achieve state-of-the-art sampling efficiency in MuJoCo motion tasks. We have shown that the linear policy matches the performance of a complex neural network policy and can be found with a simple algorithm [35][36]. We have observed that the simplicity of the algorithms and guidelines allows us to perform extensive sensitivity studies and that our method can often find good solutions to strongly non-convex problems. We have shown that direct strategies can coordinate the execution of complex nervous system strategies and can be found with simple calculations [37]. Since it is based on calculations and strategies, we have the opportunity to engage in a wide range of suggestive reasoning, and that our method can find excellent answers to very non-convex problems for a very long time to discover [38][39][40].

Our results emphasize the high variance intrinsic to the training of policies for Mujoco RL tasks. Evaluation on small numbers of random seeds does not capture performance adequately due to high variance.

REFERENCES

[1] Vassilios Tsounis, Mitija Alge, Joonha Lee, DeepGait: Planning and Control of Quadrupedal Gaits using Deep Reinforcement learning, 2019.
[2] Ziyi Zhang, Samuel Micah Akai-Nettey, Adonai Addo, Chris Rogers, Jivko Sinapov ,An Augmented Reality Platform for Introducing Reinforcement Learning to K-12 Students with Robots, 2021
[3] Octavio Villarreal, Victor Barasuol, Patrick M. Wensing, Darwin G. Caldwell, Claudio Semini ,MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion, https://doi.org/10.48550/arXiv.1909.13842 2019.
[4] Angelo Bratta, Romeo Orsolino, Michele Focchi , Victor Barasoul, On the Hardware Feasibility of Nonlinear Trajectory Optimization for Legged Locomotion based on a Simplified Dynamics, https://doi.org/10.48550/arXiv.1910.06855, 2019.
[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, Playing Atari with Deep Reinforcement Learning, 2013.
[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine, Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, 2018.
[7] Alexander L. Mitchell, Wolfgang Merkt, Mathieu Geisert, Siddhant Gangapurwala, Martin Engelcke, Oiwi Parker Jones, Ioannis Havoutis, Ingmar Posner, VAE-Loco: Versatile Quadruped Locomotion by Learning a Disentangled Gait Representation, 2022.
[8] A. Agarwal, O. Dekel, and L. Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In COLT, pages 28–40. Citeseer, 2010
[9] F. Bach and V. Perchet. Highly-smooth zero-th order online optimization. In Conference on Learning Theory, pages 257–283, 2016.
[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.
[11] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. On the sample complexity of the linear quadratic regulator. arXiv preprint arXiv:1710.01688, 2017.
[12] T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. arXiv preprint arXiv:1205.4839, 2012.
[13] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In International Conference on Machine Learning, pages 1329–1338, 2016.
[14] A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, pages 385–394, 2005.
[15] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. arXiv preprint arXiv:1611.02247, 2016.
[16] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. arXiv preprint arXiv:1702.08165, 2017.
[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.
[18] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, et al. Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv:1707.02286, 2017.
[19] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. arXiv preprint arXiv:1709.06560, 2017.
[20] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. arXiv preprint arXiv:1708.04133, 2017.
[21] K. G. Jamieson, R. Nowak, and B. Recht. Query complexity of derivative-free optimization. In Advances in Neural Information Processing Systems, pages 2672–2680, 2012.
[22] S. M. Kakade. A natural policy gradient. In Advances in neural information processing systems, pages 1531–1538, 2002.
[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
[24] S. Levine and V. Koltun. Guided policy search. In International Conference on Machine Learning, 2013.
[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.
[26] J. Matyas. Random optimization. Automation and Remote control, 26(2):246–253, 1965.
[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.
[28] M. Hutter et al., "ANYmal-a highly mobile and dynamic quadrupedal robot," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2016, pp. 38–44.

[29] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” IEEE Robot. Autom. Lett., vol. 3, no. 3, pp. 1560–1567, Jul. 2018.
[30] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, “C-CROC: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multi-contact scenarios,” IEEE Trans. Robot., 2020.
[31] J. Kim, A. Alspach, and K. Yamane, “Snapbot: a reconfigurable legged robot,” in Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017.
[32] Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, “Robust Rough-Terrain Locomotion with a Quadrupedal Robot,” in IEEE International Conference on Robotics and Automation (ICRA), 2018.
[33] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, “Feasible Region: an Actuation-Aware Extension of the Support Region,” arXiv preprint arXiv: 1903.07999, 2019.
[34] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, “Learning Agile and Dynamic Motor Skills for Legged Robots,” CoRR, vol. abs/1901.08652, 2019.
[35] R. Budhiraja, J. Carpentier, and N. Mansard, “Dynamics Consensus between Centroidal and Whole-Body Models for Locomotion of Legged Robots,” in IEEE International Conference on Robotics and Automation, may 2019.
[36] F. Doshi, E. Brunskill, A. Shkolnik, T. Kollar, K. Rohanimanesh, R. Tedrake, and N. Roy, “Collision detection in legged locomotion using supervised learning,” in IEEE International Conference on Intelligent Robots and Systems, 2007, pp. 317–322.
[37] D. Belter and P. Skrzypczynski, “Rough terrain mapping and classi- ´ fication for foothold selection in a walking robot,” Journal of Field Robotics, vol. 28, no. 4, pp. 497–528, 2011.
[38] Y. Lin, B. Ponton, L. Righetti, and D. Berenson, “Efficient Humanoid Contact Planning using Learned Centroidal Dynamics Prediction,” in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 5280–5286 .
[39] M. Giftthaler, M. Neunert, M. Stuble, M. Frigerio, C. Semini, and J. Buchli, “Automatic differentiation of rigid body dynamics for optimal control and estimation,” Advanced Robotics, vol. 31, no. 22, pp. 1225–1237, 2017.
[40] G. Bledt and S. Kim, “Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nov 2019, pp. 6316–6323.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY