Revisiting On-Policy Deep Reinforcement Learning

Mahdi Kallel[1]*, **Samuele Tosatto**[2], **Carlo D'Eramo**[1,3,4]
[1]Center for Artificial Intelligence and Data Science, University of Würzburg, Germany
[2]Department of Computer Science and Digital Science Center, University of Innsbruck, Austria
[3]Department of Computer Science, TU Darmstadt, Germany
[4]Hessian Center for Artificial Intelligence (Hessian.ai), Germany

Abstract

On-policy Reinforcement Learning (RL) offers several desirable properties, including more stable learning, less frequent policy changes, and the capacity to evaluate a policy's return during the learning process. Despite the considerable success of recent off-policy methods, their on-policy counterparts continue to lag in terms of asymptotic performance and sample efficiency. Proximal Policy Optimization (PPO) remains the de facto standard, despite its complexity and demonstrated sensitivity to hyperparameters. In this work, we introduce On-Policy Soft Actor-Critic (ON-SAC), a methodical adaptation of the Soft Actor-Critic (SAC) algorithm tailored for the on-policy setting. Our approach begins with the observation that the current on-policy algorithms do not use true on-policy gradients. We build on this observation to offer founded remedies for this problem. Our algorithm establishes a new state-of-the-art for deep on-policy RL, while simplifying the process by eliminating the need for trust-region methods and intricate critic learning schemes.

1 Introduction

A well-known dichotomy in Reinforcement Learning (RL) is the categorization in on-policy and off-policy methods [Sutton and Barto, 2018]. The former, including Proximal Policy Optimization (PPO) [Schulman et al., 2017] and Trust-Region Policy Optimization (TRPO) [Schulman et al., 2015], update the policy based on data retrieved from the current policy. In contrast, off-policy methods utilize all available data for actor updates. On-policy RL offers advantages, such as stable learning, less frequent policy changes, and the ability to assess a policy's return during the learning process.

Despite the significant achievements of recent off-policy methods like Soft Actor-Critic (SAC) [Haarnoja et al., 2018] and Twin Delayed Deep Deterministic Policy Gradients (TD3) [Fujimoto et al., 2018], their on-policy counterparts have not kept pace in terms of asymptotic performance and sample efficiency. PPO is the prevailing standard for on-policy RL and has yielded remarkable results across various domains. However, its complexity due to intricate mechanisms like trust-region optimization, multiple loss functions, and code-level optimizations, makes it highly sensitive to hyperparameter tuning [Andrychowicz et al., 2020, Huang et al., 2022].

Conversely, recent research indicates that maximum entropy RL enhances the convergence of policy gradient methods [Mei et al., 2020, Cen et al., 2024]. Despite theoretical backing, its practical implementation in on-policy deep RL algorithms remains largely unexplored. This research gap, coupled with the complexity of existing on-policy deep RL algorithms, motivates our exploration of simpler, more sample-efficient alternatives.

In this paper, we present ON-SAC, a methodical adaptation of the Soft Actor-Critic (SAC) algorithm [Haarnoja et al., 2018] to on-policy RL. Starting from the observation that existing on-policy algorithms employ biased estimates of the policy gradient, we propose to mitigate this bias by discounting

17th European Workshop on Reinforcement Learning (EWRL 2024).

*Correspondence to `mahdi.kallel@uni-wuerzburg.de`.

the policy gradient and dropping the use of momentum for the actor update. Moreover, we leverage the off-policy data for training the critic. The resulting algorithm achieves state-of-the-art performance for on-policy methods while maintaining a simple, trick-free implementation that adheres more closely to the theoretical grounding of on-policy RL.

2 Background

Reinforcement Learning deals with the problem of an agent interacting with an environment to learn a policy that maximizes its long-term reward. Mathematically, an RL problem can be formulated as a Markov Decision Process (MDP), which is a tuple (S, A, P, R, γ), where $S \in \mathbb{R}^m$ is a continuous set of states and $A \in \mathbb{R}^d$ is a continuous set of actions. $P : S \times A \times S \to [0, 1]$ is the transition probability function, where $P(s'|s, a)$ denotes the probability of transitioning to state s' after taking action a in state s. $R : S \times A \to \mathbb{R}$ is the reward function, where $R(s, a)$ is the immediate reward received by the agent for taking action a in state s. $\gamma \in [0, 1)$ is the discount factor, which determines the importance of future rewards compared to immediate rewards.

The agent's goal in an RL problem is to learn a policy $\pi : S \to \Delta A$, that maximizes its expected discounted return $G^\pi(s) = \mathbb{E}_{s\sim d^\pi_\gamma, a\sim\pi}[R(s, a)]$. Where we denote $d^\pi_\gamma(s) \triangleq \sum_{t=0}^{\infty} \gamma^t P(s_t = s)$ the improper discounted state visitation density of the state s under the policy π.

2.1 Maximum Entropy Reinforcement Learning

Traditional RL algorithms often focus solely on maximizing the expected reward. However, this can lead to overly deterministic policies that may not be robust to unforeseen changes in the environment. Maximum entropy RL [Ziebart, 2010, Haarnoja et al., 2018] addresses this issue by incorporating an entropy bonus into the objective function. The entropy of a policy π is a measure of its diversity or randomness and is defined as $H(\pi) = -\sum_{s\in S} \pi(s) \log \pi(s)$. By maximizing the entropy along with the expected reward, the agent is encouraged to explore different actions and states, leading to more robust and adaptable policies. This is achieved by introducing a Lagrange multiplier α and reformulating the objective function as:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) + \alpha H(\pi(s_t)) \right) \right] \tag{1}$$

The Lagrange multiplier α controls the trade-off between maximizing reward and maximizing entropy. A larger α leads to a greater emphasis on exploration and diversity in the policy.

2.2 Actor-Critic Methods

Actor-critic methods are a class of RL algorithms that combine two components: an actor and a critic. The critic estimates the value function, which could be the long-term expected rewards or state-value function $V^\pi(s)$, or the action-value function $Q^\pi(s, a)$. Its role is to evaluate the current policy by computing the value function of the current state. The actor updates the policy in the direction suggested by the critic by computing the policy gradient, which adjusts the policy parameters to maximize the expected return.

In essence, the critic informs the actor about the quality of its actions, allowing the actor to adjust its policy accordingly. This dual mechanism enables actor-critic methods to learn more efficiently than methods that rely solely on Monte Carlo estimates of the value function. Actor-critic algorithms improve an explicit parametric model of the critic and policy, typically implemented using neural networks, via gradient ascent. Temporal Difference (TD) learning, as described by Sutton and Barto [2018], provides an iterative method to estimate the action-value function Q^π of a policy π. The TD error is given by:

$$\delta_t = R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) - Q^\pi(S_t, A_t) \tag{2}$$

Here, R_{t+1} is the reward after transition, γ is the discount factor, and S_t, A_t and S_{t+1}, A_{t+1} are the current and next state-action pairs, respectively. The action-value function Q^π is updated to minimize the TD error, allowing the action-value estimates to be updated based on the difference between the estimated value of the next state-action pair and the current state-action pair.

The TD error δ_t serves as a learning signal that guides the update of the action-value function, a key component of many RL algorithms, including Q-learning and SARSA. The policy gradient theorem [Sutton and Barto, 2018] provides an analytical solution to the gradient of the policy return as a function of the actor's parameters:

$$\nabla_\phi J^\pi(\phi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\gamma^\pi, a\sim\pi}\left[Q^{\pi_\phi}(s,a)\nabla_\phi \log \pi_\phi(a|s)\right] \tag{3}$$

Here, d_γ^π is the discounted state distribution induced by the policy π, and Q^π is the true Q-function. In the current state of affairs, most practical algorithms propose estimators that introduce some bias in the policy improvement [Schulman et al., 2017, Haarnoja et al., 2018, Fujimoto et al., 2018]. In this work, we propose a policy gradient estimator that mitigates these bias sources.

3 On-Policy Soft Actor-critic

The current state of deep on-policy RL methods motivates us to revisit their fundamental issues and investigate the extent to which current practices align with their theoretical counterparts. In this section, we introduce and discuss *three* methodological changes for deep on-policy RL, which leads us to the introduction of our novel algorithm for deep on-policy RL, which we call *On-Policy Soft Actor-Critic (ON-SAC)* (Algorithm 1). With the introduction of this new algorithm, our goal is to construct an approach that closely adheres to the theoretical foundations of on-policy RL.

3.1 Off-Policy Critic Learning

Temporal Difference (TD) learning, as outlined by Sutton and Barto [2018], offers a methodology for learning the value function using only system transitions, as expressed in Equation equation 2. This algorithm is a cornerstone in the field of RL, with extensive research dedicated to understanding its properties. It is well known that when TD is applied to a tabular value function representation, it converges to the true value function [Dayan, 1992, Jaakkola et al., 1993]. On the other hand, on-policy TD learning approaches using linear function approximation have been proven to converge to a fixed point in the vicinity of the projection of the true value function [Tsitsiklis and Van Roy, 1996]. However, divergence may occur with standard TD learning when states are sampled off-policy and linear function approximation is used [Baird, 1995]. This issue has prompted the creation of several alternative algorithms specifically engineered to guarantee convergence under off-policy sampling [Kolter, 2011, Diddigi et al., 2019]. In light of the lack of convergence guarantees for simple off-policy TD learning, the desirable properties of on-policy TD learning have inspired the development of deep RL on-policy methods that learn using exclusively freshly generated data, forgoing the use of a replay buffer to store previous transitions [Schulman et al., 2015, 2017].

Algorithm 1 One step of ON-SAC

```
Require: $\phi$, $\theta_1$,.., $\theta_K$, $\mathcal{B}$, $\gamma_t = 1$                ▷ Initial parameters
  $\mathcal{D} \leftarrow \emptyset$                                    ▷ Reset the actor replay buffer
  for $N_e$ episodes do
      for each environment step do
          $a_t \sim \pi_\phi(a_t|s_t)$                              ▷ Sample action from the policy
          $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$                 ▷ Sample transition from the environment
          $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, r_t, s_{t+1}, \gamma_t\}$          ▷ Update the actor replay buffer
          $\mathcal{B} \leftarrow \mathcal{B} \cup \{s_t, a_t, r_t, s_{t+1}\}$              ▷ Update the critic replay buffer
          $\gamma_t \leftarrow \gamma_t \times \gamma$                              ▷ Update the discount factor
      end for
      $\gamma_t \leftarrow 1$
  end for
  for $N_u$ update steps do
      $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, .., K\}$     ▷ Update the $Q$-function parameters
  end for
  $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$                           ▷ Update policy weights
  $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$                             ▷ Adjust temperature
Ensure: $\phi$, $\theta_1$,.., $\theta_K$                                ▷ Optimized parameters
```

When looking closer, one of the primary factor contributing to this non-convergence is state aliasing, a phenomenon that occurs in off-policy approximation [Sutton et al., 2016] when the function approximator perceives different states as identical, leading to information loss and potential divergence in learning. Theoretically, the bias introduced by off-policy approximation diminishes with larger regressors [Sutton et al., 2016]. This is attributed to the ability of larger regressors to capture more nuances in the state representation, thereby reducing the likelihood of state aliasing. However, it is crucial to note that while larger regressors can mitigate bias, they may concurrently increase the variance of the estimates. By using this insight, we chose to integrate off-policy data into our critic learning scheme by keeping track of past transitions via a replay buffer.

3.2 Properly Discounting The Policy Gradient

The policy gradient theorem, as defined in Equation equation 3, describes the gradient of the anticipated discounted return in relation to an agent's policy parameters. Despite this, the majority of policy gradient methods bypass the use of the discounted state distribution, denoted as d_γ^π, when calculating the policy gradient, opting to average the gradients across states instead. However, this practice results in a biased gradient estimator as it fails to optimize the discounted objective.

Research has shown that this averaged gradient does not embody the gradient of any function [Nota and Thomas, 2019]. As a result, there is no guarantee that algorithms following this direction will converge to a 'reasonable' optimum. In fact, it is possible to construct a counterexample where the fixed point is globally pessimal for both the discounted and undiscounted objectives [Nota and Thomas, 2019]. Despite these shortcomings, this estimator remains the most widely used for estimating the policy gradient, primarily due to its proven effectiveness in practical applications [Schulman et al., 2017, Haarnoja et al., 2018, Fujimoto et al., 2018]. Hence, to stay true to the theory of reinforcement learning, and to make sure that we are optimizing for a valid objective, we use the discounted state distribution d_γ^π for our policy gradient.

3.3 Using Pure Gradient Ascent

RMSProp (Root Mean Square Prop) is an adaptive learning rate optimization algorithm designed to address the shortcomings of vanilla stochastic gradient descent (SGD) in neural network training [Hinton and Swersky, 2012]. SGD can struggle with diminishing or exploding gradients, hindering convergence. RMSProp tackles this issue by maintaining a moving average of the squared gradients for each parameter. This average is used to normalize the gradients, leading to larger updates for infrequently changing parameters and smaller updates for those with high gradient variance. This moving average of squared gradients is an approximation of the diagonal elements of the Hessian that does not capture the full curvature information. However, compared to second-order optimization methods, RMSProp avoids the computational cost of calculating the Hessian matrix, making it more efficient for large-scale problems.

Adam (Adaptive Moment Estimation) builds upon RMSProp by incorporating an additional moving average of the gradients themselves[Kingma and Ba, 2014]. A standard in all deep learning algorithms, Adam has imposed itself as the defacto optimizer for deep reinforcement learning methods [Schulman et al., 2017, Haarnoja et al., 2018, Fujimoto et al., 2018]. This momentum makes our current policy gradient take into account the previously accumulated policy gradients. For an order of magnitude, with the default momentum of $b_1 = 0.9$, our current gradient at timestep t is contributing only $\frac{1}{1-b_1} = 10\%$ to the current gradient estimate. This makes us question the true "on-policyness" of such gradients. With this insight in mind, we use RMSProp as our optimizer to capitalize on the faster convergence properties while avoiding bias in the gradient.

4 Results

By taking into account the remarks made above, we empirically evaluate our algorithm ON-SAC against PPO. We highlight the main differences between our algorithm and PPO in Table 1. For our implementation, we use an ensemble of 5 critics trained independently and we rollout the policy for 5 episodes for each policy update step. We also use a discount factor of 0.995 and we halve the survival reward to avoid local optima. Indeed, when using discounted gradients the undiscounted performance becomes more sensitive to the discount factor as we are truly optimizing for the discounted objective.

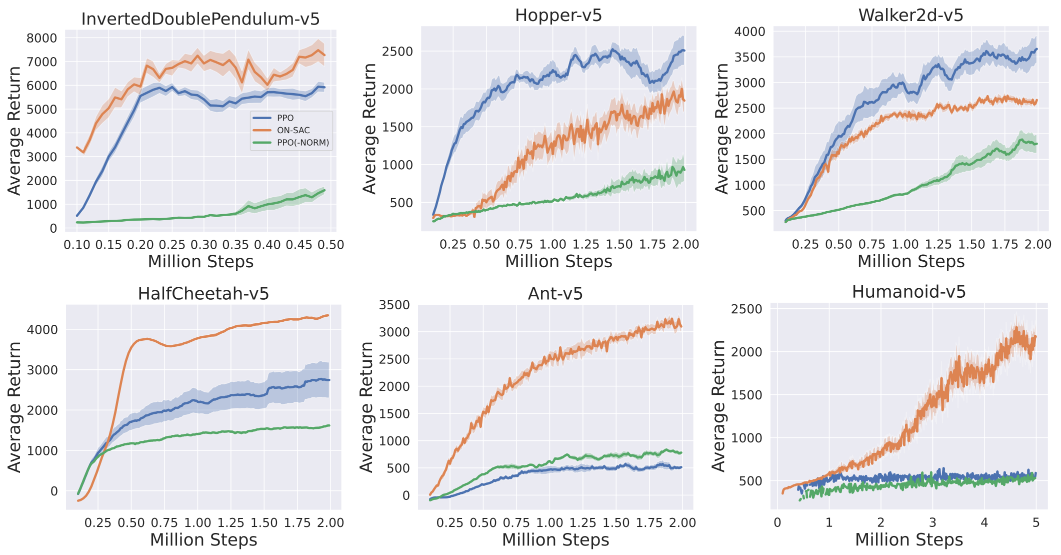

Figure 1: Evolution of the undiscounted policy return on the MuJoCo-v5 tasks. We use 10 random seeds for every algorithm.

In Figure 1, we see that our algorithm ON-SAC outperforms PPO on 4 out of 6 MuJoCo tasks with a considerable performance gap for higher dimensional problems. We also demonstrate that removing just one code-level optimization, in our case, the reward normalization step makes our algorithm outperform PPO by a large margin on all tasks. This is in contrast to our algorithm which uses no code-level optimizations. In Figure 2, we report the results the standard discount factor of $\gamma = 0.99$.

In Figure 3, we conduct an ablation study on our algorithm, focusing on three primary design choices: the use of off-policy Temporal Difference (TD) learning, the application of the true discounted policy gradient, and the absence of momentum during optimization.

Our findings indicate that using momentum, for example Adam with $b1 = 0.99$, deteriorates performance across all tasks, with the sole exception being the InvertedDoublePendulum task. We believe this is because Adam, and hence momentum, relies on loss functions and neural networks that

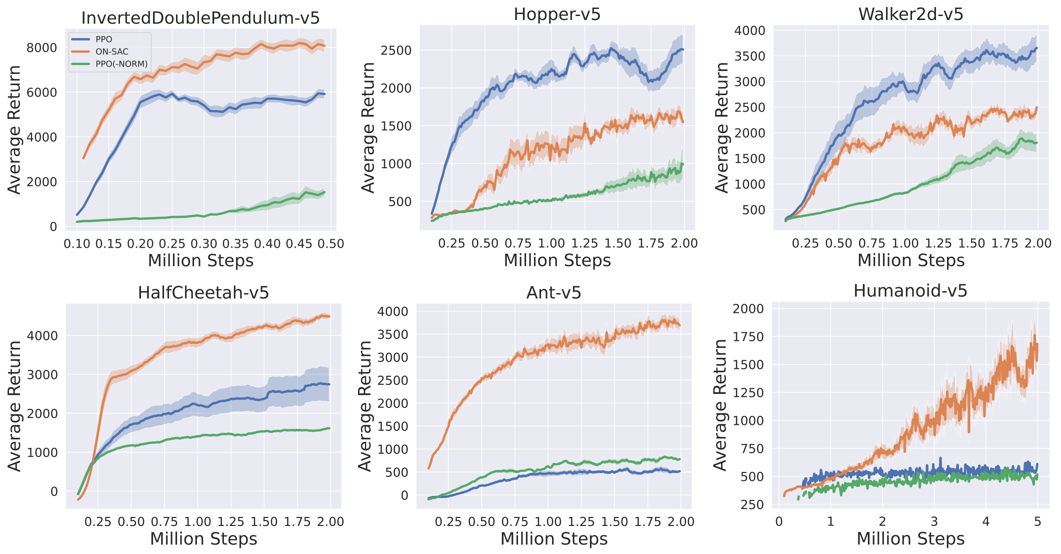

Figure 2: Evolution of the policy return on the MuJoCo-v5 tasks for various design choices. We use 10 random seeds for every algorithm.

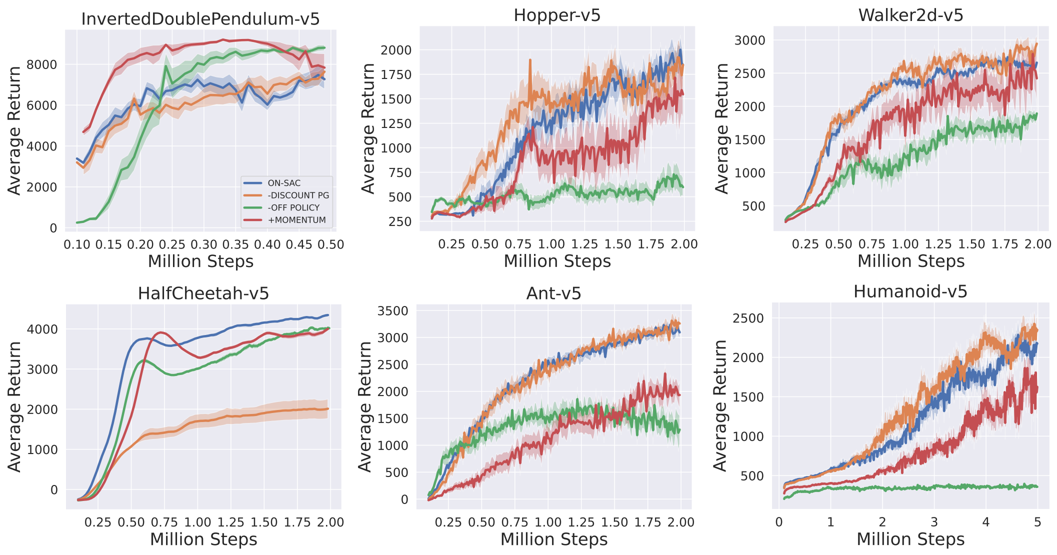

Figure 3: Evolution of the policy return on the MuJoCo-v5 tasks for various design choices. We use 10 random seeds for every algorithm.

are analytically differentiables, as common in supervised and unsupervised learning. However, in RL, the gradient of the objective ∇J^{π} relies on the critic's estimate, which introduces an additional bias in the gradient approximation. While in supervised and unsupervised learning, the gradient is unbiased, in RL, the use of function approximation for the critic induces bias in the gradient estimate. We hypothesize that adding momentum to the optimizer may reinforce this bias, leading to detrimental effects on policy improvement.

Similarly, our experimental results indicate that restricting the training of the critic to on-policy data significantly affects performance and can even hinder learning as in the case of the Humanoid-v5 task. The only exception is the InvertedPendulum-v5 task which we believe did not suffer as it requires less data to learn a critic for a small dimensional space. In general, we observe that training the critic on larger datasets, even when off-policy, proves to be beneficial compared to limiting the training to the small pool of freshly generated data. As for the discounted policy gradient, we observe that employing a discount on the policy gradient has minimal impact on the policy performance, benefiting the HalfCheetah task where it leads to a significant improvement. We believe this task resembles the synthetic task formulated in [Nota and Thomas, 2019], where the averaged policy gradient results in a degenerate local optimum.

Our study presents a contrasting perspective on the impact of using momentum in PG, diverging from the conclusions drawn by Andrychowicz et al. [2020], who found that alterations in optimizers have minimal influence on the performance. We attribute this discrepancy to the superior quality of the gradient employed in our approach. Unlike PPO, which utilizes mini-batches of on-policy data for actor updates, our method implements a single gradient descent step using the full batch. This modification, in conjunction with the aforementioned enhancements, improves our policy gradient estimate, aligning it more closely with the true policy gradient. In such a scenario, we theorize that momentum could be counterproductive due to the inherently non-stationary characteristics of RL.

Attribute	**PPO**	**ON-SAC**
Trust region	✓	-
Complicated critic	✓	-
Code level optimization	✓	-
Discounted policy gradient	-	✓
Is truly on-policy	-	✓
Uses off-policy data	-	✓

Table 1: Comparison of PPO and Supersac

In summary, our results suggest that off-policy TD learning consistently enhances performance. Conversely, the use of momentum appears to introduce significant bias, undermining the quality of the policy gradient. Lastly, the discounting of the gradient contributes to a more robust algorithm.

5 Related works

Entropy regularization plays a pivotal role in numerous practical deep reinforcement learning (RL) algorithms. The entropy of the policy, in fact, acts as a regularizer, shaping the objective landscape [Ahmed et al., 2019]. One prevalent strategy for incorporating entropy regularization into Policy Gradient (PG) methods involves the addition of an entropy cost term to the sample-based policy gradient estimator. This approach aims to maximize the policy entropy at each sampled state, thereby preserving the policy's stochasticity throughout the optimization process [Mnih et al., 2016, Schulman et al., 2017]. An alternative strategy regularizes the policy evaluation phase by supplementing the standard RL task objective with an entropy term. This method guides policies towards regions of higher expected trajectory entropy, a scheme often referred to as Maximum Entropy RL (MaxEnt RL) [Ziebart, 2010, Haarnoja et al., 2018]. The MaxEnt RL formulation is recognized for enhancing the exploration capabilities and robustness of policies by fostering stochasticity. In practical terms, MaxEnt RL can be implemented simply by appending an entropy reward to the original task reward. While the entropy bonus term strives to maximize policy entropy at visited states, MaxEnt RL steers a policy towards regions of higher expected trajectory entropy. However, this comes with the trade-off of introducing bias into the objective [Schulman et al., 2017]. Recent studies on PG methods have underscored the efficacy of MaxEnt RL in expediting the convergence of Policy Gradient methods [Mei et al., 2020, Ahmed et al., 2019, Cen et al., 2024].

In a study by Ilyas et al. [2018], it was observed that the behavior of deep policy gradient algorithms differs greatly from its motivating frameworks. Specifically, learned value estimators frequently fail to fit the true value function, and there is a poor correlation between gradient estimates and the 'true' gradient. In Nota and Thomas [2019], the authors demonstrated that the undiscounted policy gradient does not correspond to the gradient of any objective function. They also identified instances where this empirical gradient can be suboptimal for both discounted and undiscounted policy return. Moreover, numerous studies have underscored the sensitivity of current deep on-policy methods to hyperparameters and implementation details [Huang et al., 2022, Andrychowicz et al., 2020].

In their study, Henderson et al. [2018] demonstrate that, when applied to RL tasks, the effectiveness of adding momentum to adaptive optimizers varies with every environment. This variation is linked to the fact that in online stochastic settings, which is the case of RL, adaptive optimizers like Adam and RMSProp can fail to converge to an optimal solution [Reddi et al., 2019, Wilson et al., 2017].

Temporal Difference (TD) learning Sutton and Barto [2018], is a key RL algorithm for learning value functions using system transitions. It is known to converge to the true value function when applied to a tabular value function representation [Dayan, 1992, Jaakkola et al., 1993]. On-policy TD learning with linear function approximation converges to a fixed point near the true value function's projection [Tsitsiklis and Van Roy, 1996]. However, divergence can occur with off-policy state sampling and linear function approximation Baird [1995], leading to the development of alternative algorithms ensuring convergence under off-policy sampling [Kolter, 2011, Diddigi et al., 2019].

6 Conclusion

In this work, we proposed ON-SAC, an adaptation of the Soft Actor-Critic algorithm specifically tailored for on-policy Reinforcement Learning (RL). ON-SAC stands as a fundamental benchmark for on-policy methodologies, faithfully adhering to the theoretical foundations of on-policy RL while maintaining a clean, trick-free implementation. Our algorithm focuses on the quality of the policy gradient. This allows it to circumvent the necessity for trust regions and intricate critic learning schemes. Moreover, ON-SAC delivers state-of-the-art performance for deep on-policy and defines a robust and uncomplicated baseline for future deep on-policy methods. We believe that ON-SAC paves the way for further exploration in the realm of on-policy RL, providing a simple and solid foundation upon which future research can build upon.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) (Project: 01IS22078). The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b187cb. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683.

References

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pages 151–160. PMLR, 2019.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*, 2020.

Leemon Baird. Residual algorithms: reinforcement learning with function approximation. In *machine learning proceedings 1995*, pages 30–37. Elsevier, 1995.

Shicong Cen, Yuting Wei, and Yuejie Chi. Fast policy extragradient methods for competitive games with entropy regularization. *Journal of machine learning Research*, 25(4):1–48, 2024.

Peter Dayan. The convergence of td (λ) for general λ. *machine learning*, 8:341–362, 1992.

Raghuram Bharadwaj Diddigi, Chandramouli Kamanchi, and Shalabh Bhatnagar. A convergent off-policy temporal difference algorithm. *arXiv preprint arXiv:1911.05697*, 2019.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Peter Henderson, Joshua Romoff, and Joelle Pineau. Where did my optimum go?: An empirical analysis of gradient descent optimization in policy gradient methods. *arXiv preprint arXiv:1810.02525*, 2018.

Srivastava Hinton and Swersky. Overview of mini-batch gradient descent. 2012.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. *arXiv preprint arXiv:1811.02553*, 2018.

Tommi Jaakkola, Michael Jordan, and Satinder Singh. Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems*, 6, 1993.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J Kolter. The fixed points of off-policy td. *Advances in neural information processing systems*, 24, 2011.

Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International conference on machine learning*, pages 6820–6829. PMLR, 2020.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on mAcHiNe lEaRnInG*, pages 1928–1937. PMLR, 2016.

Chris Nota and Philip S Thomas. Is the policy gradient a gradient? *arXiv preprint arXiv:1906.07073*, 2019.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of machine learning Research*, 17(73):1–29, 2016.

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A Hyperparameters

Table 2 lists of the hyperparameters used for ON-SAC.

Table 2: ON-SAC Hyperparameters

Parameter	Value
optimizer	RMSProp [Hinton and Swersky, 2012]
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.995
replay buffer size	10^5
number of critics	5
number of hidden layers (all networks)	2
number of hidden units per layer	256
number of samples per minibatch	256
entropy target	$-\dim(\mathcal{A})$
nonlinearity	TanH
actor update interval	5 episodes