The Cliff of Overcommitment with Policy Gradient Step Sizes

Scott M. Jordan[*,1,2], **Samuel Neumann**[*,1,2], **James E. Kostas**[3]
Adam White[1,2,4], **Philip S. Thomas**[3]
{sjordan, sfneuman, amw8}@ualberta.ca, {jekostas, pthomas}@cs.umass.edu
[1]University of Alberta, Department of Computing Sciences
[2]Alberta Machine Intelligence Institute
[3]University of Massachusetts, College of Information and Computer Sciences
[4]Canada CIFAR AI Chair

Abstract

Policy gradient methods form the basis for many successful reinforcement learning algorithms, but their success depends heavily on selecting an appropriate step size and many other hyperparameters. While many adaptive step size methods exist, none are both free of hyperparameter tuning and able to converge quickly to an optimal policy. It is unclear why these methods are insufficient, so we aim to uncover what needs to be addressed to make an effective adaptive step size for policy gradient methods. Through extensive empirical investigation, the results reveal that when the step size is above optimal, the policy overcommits to sub-optimal actions leading to longer training times. These findings suggest the need for a new kind of policy optimization that can prevent or recover from entropy collapses.

1 Introduction

Reinforcement learning (RL) algorithms, like any optimization software, should reliably solve many problems without requiring the user to understand how the algorithms work. However, RL algorithms are not easy to apply because they require carefully selection of hyperparameters (e.g., step size and policy structure), which often requires expert knowledge of both the algorithm and application. Furthermore, domain experts, not just RL experts, will apply the algorithms as RL's use becomes prevalent. Thus, it is crucial to design algorithms that non-experts can easily and reliably use.

Policy gradient methods, which aim to approximate optimal policies using stochastic gradient ascent, are of particular interest because they can be used in continuous-action control tasks common in industry. The effectiveness of these algorithms depends on many factors, but a particularly critical one is the choice of step size. There are numerous policy gradient algorithms, many of which include an adaptive step size component to address this sensitivity. For example, natural policy gradient algorithms (Kakade, 2002; Morimura et al., 2005; Peters & Schaal, 2008) adapt the step size to account for the parameterization of the policy and enable the policy to change quickly in regions where the the policy is nearly deterministic. Many algorithms (Schulman et al., 2017; Mnih et al., 2016; Henderson et al., 2018) use optimizers such as RMSProp or Adam from stochastic optimization that adapt the step sizes based on statistics of the gradient estimate, e.g., the norm or variance (Hinton et al., 2012; Kingma & Ba, 2015; Mei et al., 2021b). Some methods compute an upper bound on the step size such that policy improvement is guaranteed with high probability (Pirotta et al., 2013; Papini et al., 2019) or use line search to estimate the optimal step size for each

[*]These authors contributed equally to this work.

update (Schulman et al., 2015). Recent work has focused on optimizing the step size during learning (Paul et al., 2019; Jaderberg et al., 2017), but these methods introduce additional hyperparameters. Despite all these efforts, no method exists that quickly finds a good policy and does not require tuning. So instead of trying to create yet another policy gradient optimizer, the goal of this study is to further the understanding of step sizes and how they impact the performance of policy gradient methods.

An effective adaptive step size method will make it so the algorithm can quickly and reliably converge to a high-performing policy with a wide range of initial step sizes. In this paper, we seek insights into what key obstacles still need to be addressed to develop an effective adaptive step size strategy. More specifically, we want to know how the behavior of the algorithm changes when the step size is around the optimal value and why these rescaling methods are failing. We show that these failures occur when the step size is above the optimal value and the entropy of the policy collapses too quickly. In turn, this results in a lack of exploration and the policy is trapped in sub-optimal behavior for a long time. We call this point where entropy drops to quickly, the cliff of overcommitment.

2 Background

In this section, we provide background on RL and define the notation used in this paper. We represent the environment an agent interacts with as a finite and episodic Markov decision process (MDP). An MDP, M, is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, d_0, \gamma)$, where $\mathcal{S}$ is the set of all states, $\mathcal{A}$ is the set of all actions. At times $t \in \{0, 1, \dots, T-1\}$ the agent is in state S_t, selects an action A_t, receives a reward R_t, and transitions to state S_{t+1}, the reward function $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines the expected reward an agent receives for being in state $s \in \mathcal{S}$ and taking action $a \in \mathcal{A}$, i.e., $r(s,a) \coloneqq \mathbf{E}[R_t | S_t = s, A_t = a]$, the transition function $p\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ specifies the probability of an agent entering state $s' \in \mathcal{S}$ after taking action a in state s, i.e., $p(s,a,s') \coloneqq \Pr(S_{t+1} = s' | S_t = s, A_t = a)$, the initial state distribution is defined through the function $d_0\colon \mathcal{S} \to \mathbb{R}$, such that $d_0(s) \coloneqq \Pr(S_0 = s)$, and $\gamma \in [0,1]$ is a reward discount parameter. In this paper, we use $\gamma = 1.0$. We refer to the sequence $s_0, a_0, r_0, \dots, s_{T-1}, a_{T-1}, r_{T-1}$ as an episode.

We call the mechanism an agent uses to select an action a *policy*, and represent it with the function $\pi\colon \mathcal{S} \times \mathcal{A} \to [0,1]$, such that $\pi(s,a) \coloneqq \Pr(A_t = a | S_t = s)$. In this paper we focus on parameterized policies, $\pi\colon \mathcal{S} \times \mathcal{A} \times \mathbb{R}^n \to [0,1]$, where the policy takes as additional input parameters $\theta \in \mathbb{R}^n$ to specify the conditional distribution over actions. An agent's objective is to find policy parameters $\theta^\star$ that approximately maximize $\rho(\theta) \coloneqq \mathbf{E}[G]$, where $G = \sum_{t=0}^{T-1} \gamma^t R_t$.

A common method to search for $\theta^\star$ is to use gradient ascent to update the parameters iteratively, i.e., $\theta \leftarrow \theta + \eta \nabla \rho(\theta)$, where $\eta > 0$ is a step size. The policy gradient theorem (Sutton et al., 2000) provides an expression for the gradient, but the algorithms we study will derive approximations from the form: $\nabla \rho(\theta) \coloneqq \mathbf{E}[G\Psi]$, where $\Psi = \sum_{t=0}^{T-1} \frac{\partial \ln \pi(S_t, A_t, \theta)}{\partial \theta}$ (Tang & Abbeel, 2010). For completeness, we prove the equivalence of these two forms in Appendix A. One of the simplest stochastic gradient ascent (SGA) algorithms for policy gradient methods is the REINFORCE algorithm (Williams, 1992), which uses an unbiased estimate of the gradient $\widehat{\nabla} = \frac{1}{k} \sum_{i=1}^{k} (G_i - b)\Psi_i$, where k is the number of episodes to sample, G_i and Ψ_i are the samples from the i^{th} episode, and $b \in \mathbb{R}$ is a baseline and is often an estimate of $\rho(\theta)$. In this paper, we investigate the following policy update methods:

$$\begin{array}{lll} \text{SGA} & \theta \leftarrow \theta + \eta \widehat{\nabla} & \\ \text{RMSprop} & \theta \leftarrow \theta + \frac{\eta}{\sqrt{v} + \epsilon} \widehat{\nabla} & v \leftarrow v + \beta(\widehat{\nabla}^2 - v) \end{array}$$

where $\epsilon > 0$ is a regulation parameter, $\widehat{\nabla}^2$ is an elementwise squaring of the gradient estimate. We use a running average of the the scaling statistics, v, used in the step size methods, e.g., $l \leftarrow l + \beta(\|\widehat{\nabla}\| - l)$, where $\beta \in (0,1)$. For our experiments, we use $\beta = 0.05$ because it performed well enough but in no way represents an optimal choice or necessarily a reliable one for any given problem. Additionally, it is important to note that the regularization ϵ can greatly impact the policy optimization process. Using a small ϵ, e.g., $\epsilon = 10^{-8}$, can allow the step size to grow large, while a smaller ϵ, e.g., $\epsilon = 10^{-1}$

can prevent the step size from growing too large, but then the adaptive step size has little impact once ϵ is larger than the other terms. We also investigate Adam optimizer but finds that it is similar to RMSProp. We report the results for it in Appendix E

3 Policy Gradient Warm Up

The performance of policy gradient methods is sensitive to the step size because the step size directly controls the exploration-exploitation trade-off. To see why, first consider that the step size, η, controls how much change is allowed to the policy's distribution over actions, where increasing (decreasing) η leads to larger (smaller) changes in action distribution. Additionally, for some policy parameterizations, e.g., softmax, as $\eta \to \infty$, the policy becomes greedy with respect to a given action sequence (Kakade, 2002; Wagner, 2011). If a policy becomes too deterministic, then little exploration happens, and the policy will become trapped for long periods and will not improve (Schaul et al., 2019). Setting η to be small will prevent the policy from becoming too deterministic too quickly but will result in less exploitation, and improvement will be slow. So, one should select the step size to balance exploration and exploitation throughout learning.

Balancing exploration and exploitation with η is challenging because the amount a policy will change depends on the magnitude of the gradient estimate $\|\widehat{\nabla}\|$. One can interpret the policy update as a step in the direction of the unit length vector $\widehat{\nabla}/\|\widehat{\nabla}\|$, with an *effective step size* $\eta\|\widehat{\nabla}\|$. The magnitude $\|\widehat{\nabla}\|$ can vary significantly from problem to problem and during learning. To make stochastic gradient ascent methods robust to changes in the magnitude of the component of the gradients, gradient rescaling methods such RMSprop, Adam (Kingma & Ba, 2015), and return rescaling (Hafner et al., 2023) change the gradient so the amount the policy changes is more consistent. The result is that these methods often produce similar ranges of good step sizes across different problems. While this makes it easier to search for a good step size, it does not remove the need to tune the step size for each problem. Our goal is to understand when these rescaling methods will lead to a failure in learning and what an adaptive step size method will need to address.

4 Experiment Settings

There are many variations of policy gradient algorithms, however, they are all based around the REINFORCE update. So we study the basic update method first to establish our hypothesis and insights, then we will check that it remains true with PPO. Furthermore, the parameterizaton of the action distribution and function approximation can cause differences in the results. So, we investigate the combinations of softmax and squashed Gaussian (tanh of a Gaussian random variable) distributions with both linear function approximation and neural networks.

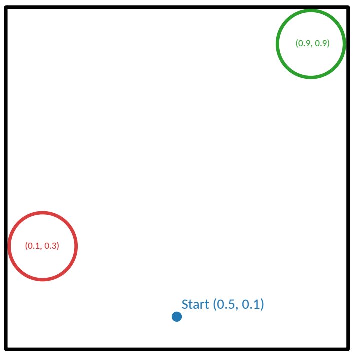

Figure 1: This is an illustration of the 2D environment. The red and green circles are goal regions and give rewards of +1 and +10, respectively.

For our experiments to be thorough, we need to be able to run the agent many times for many different step sizes and be able to identify what happens to the agent's policy. For this reason, we begin our investigation with a simple two dimensional world where the agent needs to get to a goal state as quickly as possible. To make this problem, slightly harder, we introduce a second goal state that is closer to the start state and gives a smaller reward. The agent receives a reward of -0.01 every step, a reward of $+1$ if it enters the close goal, and a reward of $+10$ if it enters the far goal. For discrete actions the agent has nine actions: each of the four cardinal directions, the four diagonals, and a no-op action. For continuous actions the can choose any pair in $[-1, 1]^2$, which leads to a displacement in each coordinate of the agent's position. For each settings the agent moves up to

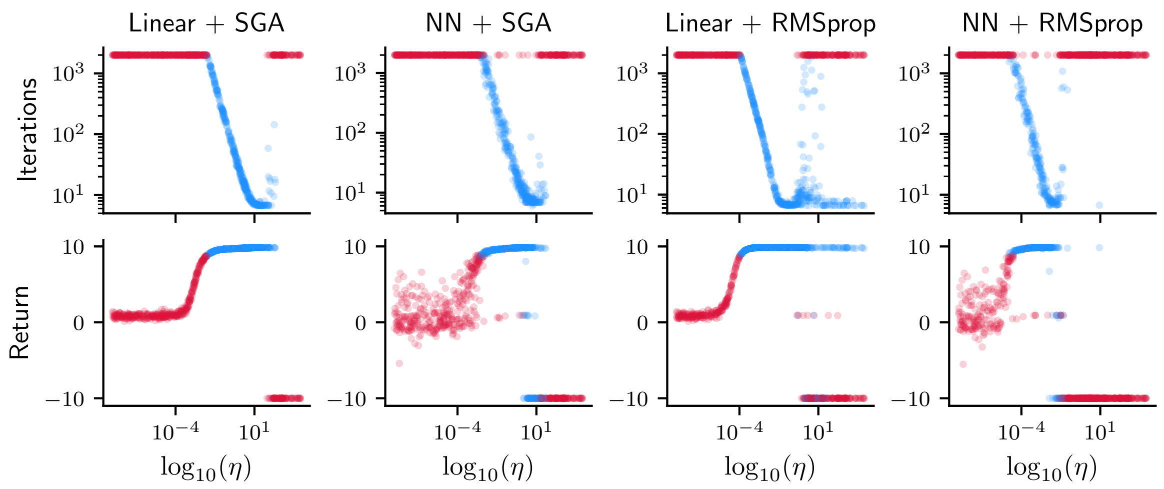

Figure 2: The top row shows the number of iterations it took for REINFORCE with a softmax policy to find a near optimal policy for each step size. The bottom row shows the averaged return (from the last 300 episodes) when the algorithm was allowed to run for 2,000 iterations. Each column represent a different combination of function approximator (linear or neural network) and optimizer. Each dot represent one run of the algorithm with a specific step size. The color of the dot indicates success (blue) or failure (red) in finding a near optimal policy. When analyzing the number of iterations, pay attention to the log scale on the vertical axis. The scaling means the time it takes to find a good policy increases very quickly for a step size that is too small. Additionally, note that some blue dots will have a final return less than the threshold, because the algorithm was allowed to run past the point that it found a near optimal policy and got worse.

0.05 units in each direction. We illustrate this environment in Figure 1. In Section 7, we test our findings on the MuJoCo Ant environment (Todorov et al., 2012).

5 Modeling Performance Sensitivity

In this section, we focus on understanding how performance changes as a function of the step size. Specially, we want to understand how performance changes for step sizes above and below the optimal step size.

Before continuing, we need to define the performance metric of interest. Two metrics categorize an optimization algorithm's performance, the quality of the final solution, and the time it takes for the algorithm to terminate. However, in RL research, the algorithms are often run for a fixed amount of time (episodes or time steps), and only the expected return of the final policy is considered, i.e., $\rho(\pi_{\text{final}})$. This paper examines something different: we run each algorithm until the policy performance is above a chosen threshold, THRESH. In this sense, we view performance as the time, in the number of episodes, it takes for the algorithm to reach the threshold. To account for stochasticity in estimating the $\rho(\pi)$ we estimate a lower bound $\rho^-(\pi)$ and terminate when $\rho^-(\pi) \geq \text{THRESH}$. We construct $\rho^-(\pi)$ using the previous n returns, i.e., $\rho^-(\pi) = \bar{G}_{i-n+1:i} - 3\frac{\sigma(G_{i-n+1:i})}{\sqrt{n}}$, where i is the current episode number and $\bar{G}_{i-n+1:i}$ and $\sigma(G_{i-n+1:i})$ are the average and standard deviation of the n most recent returns. This lower bound is similar to a confidence interval using z-scores but is biased due to data reuse. We use the constant 3 to scale the interval as it eliminated almost all false positives in initial tests. Additionally, we set an upper limit of 2,000 iterations ($N = 100{,}000$ episodes). This limit avoids the algorithm running too long, but it is high enough that we can still see the effects for a wide range of step sizes.

For 800 randomly chosen step sizes in the range $[10^{-8}, 10^4]$, we show both the number of of iterations to find a near optimal policy and the policy performance after 2,000 iterations on the 2D environment,

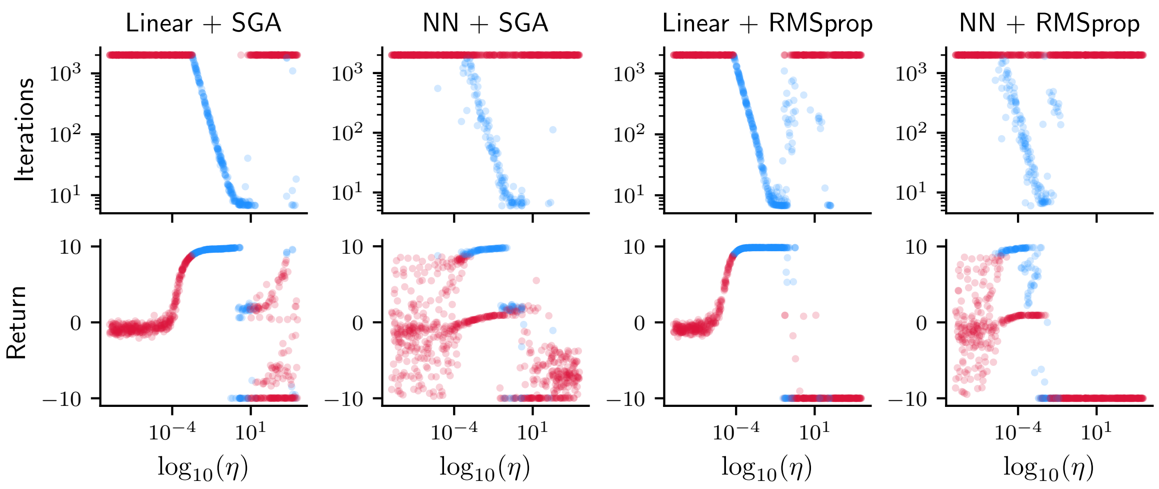

Figure 3: These are the same as those in Figure 2, but correspond to using the squashed Gaussian parameterization. The biggest difference to note is that often REINFORCE was unable to find a near optimal policy with neural network, even when using good step sizes.

in Figures 2 and 3. The first thing to notice in these plots, is that the performance is asymmetric around the optimum step size (the step size that minimizes the expected number of iterations to find a near optimal policy). Step sizes that are below the optimal step size have a high likelihood of finding a good policy. Step sizes that are above optimal quickly increase optimization time and have a reduced chance to find a near optimal policy. We refer to this phenomenon as the *cliff of overcommitment*. We investigate this further in the following sections.

6 Sensitivity around the Optimum

The performance around the optimal step size is asymmetric, with the performance above optimal step size forming a cliff, where the step size being above optimal leads to longer training times and reduced chance of finding a near optimal policy. Two things are apparent in Figures 2 and 3. The first is that the optimal step size is not always the step size that achieves the fastest convergence. The second is that the optimal step size is slightly below or just into the region where the algorithm does not consistently find a policy above the performance threshold. This finding implies that even if the variation of the optimal step size is moderately small, choosing a step size based on the center of optimal step sizes means that for some problems, the algorithm will not be likely to find a good policy. Thus, to develop reliable adaptive step size methods, we need to understand why the algorithm fails when the step size goes above the optimal value.

There are several possible reasons for the algorithms failing to find a good policy above the optimal step size, such as overstepping, where the policy changes too much, and performance decreases or diverges, as is common in supervised learning. While this can happen, especially if the gradient estimate is poor, we hypothesize a different reason: with large step sizes, the policy becomes nearly deterministic too quickly, and there is insufficient exploration to improve the policy. To test this hypothesis, we need to show two things: 1) that the step size has a direct impact on the rate at which the entropy decreases, 2) that trials that successfully find a good policy have higher entropy in the policy than the trials that fail to find a good policy, and 3) that the performance is not decreasing as step sizes go above the optimal step size.

To test this hypothesis, we run the SGA and RMSprop update methods while recording the policy's returns and entropy at each iteration. We use $H(\theta) := -\mathbf{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \ln \pi(S_t, A_t)\right]$ as the measure of policy entropy for softmax policies. For continuous actions, we discretize the action space and use the entropy of the sampled actions overall time. See Appendix C for details.

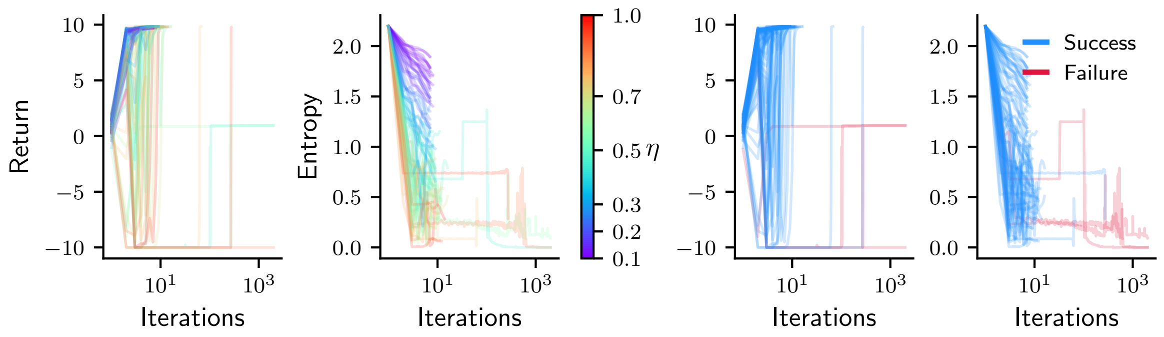

Figure 4: (left, middle-left) These plots show the return and entropy over each iteration for the REINFORCE with RMSprop and softmax policies. Each line corresponds to a run of the algorithm with randomly chosen step size and the line color indicates the step size. (middle-right,right) These plots illustrate the same data as the other two, but are colored based on if the a near optimal policy was successfully found or not. The lines are truncated to the point where a near optimal policy was found. Notice how the step size directly impacts how fast the entropy decreases. Additionally, note that the trials that go on a long time all have low entropy.

We plot the returns and entropy over time for each trial for REINFORCE with linear function approximation, RMSprop, and softmax policy parameterization in Figure 4. We show the results for other configurations in Appendix E. In Figure 4, the relationship between step size and the rate of entropy reduction is apparent: a larger step size leads to a faster reduction in entropy. When the step size is small, the rate of entropy reduction is consistent and has little variability. However, as the step size increases, so does the noise in the entropy reduction process.

To answer the second part of our hypothesis, trials that fail to find a good policy reach low entropy policies, we use Figure 4 to show the differences in successful and unsuccessful trials. The results show that the algorithm will successfully terminate with a good policy for a sufficiently slow enough drop in entropy. In some of the successful trials entropy drops quickly. This can occur if the policy prioritizes going towards the better goal state and then becomes nearly deterministic. If this does not happen, then it is unlikely, but still possible that the policy will find the near optimal goal and be able to reach it. This event can be seen in middle-right plot of Figure 4, where there are two blue spikes later in learning.

To answer the last part of our hypothesis, that the failure is due to small entropy and not due to overstepping and finding a worse policy or diverging, we visualize the returns over time in Figure 4. We see that both successful and failed trials have overstepping where performance notably gets worse. However, on the failed trials, performance is stuck in a performance plataeu and have low entropy. Decreases in performance are, unsurprisingly, more common at higher step sizes. Together, this suggests that overstepping may occur, but the policy can eventually improve if the entropy remains high enough. Thus with step sizes larger than optimal, we say that the policy overcommits to suboptimal actions and becomes less likely to improve.

This hypothesis does not explain all observations in Figures 2 and 3. Specially, that when optimizing neural network policies there are failures even when using small step sizes. This was particularly common for the squashed Gaussian policies. We discuss some plausible explanations for these failures in Section 8, but leave a precise answer to future work. Although these events are unexplained, our hypothesis is well supported and explains most of the observations.

7 Evaluating Entropy Collapse in PPO

While the experiments in the previous section illustrate the overcommitment cliff with the basic REINFORCE method in a simple environment, we need to check and make sure it is present with

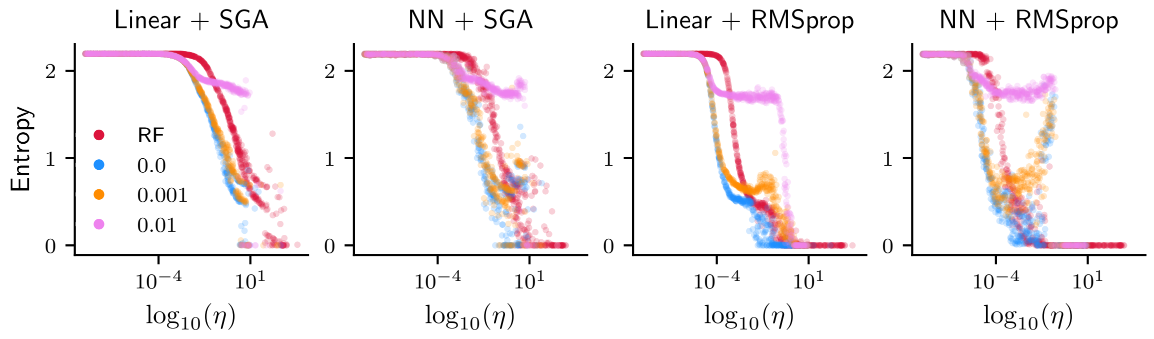

Figure 5: These plots show the final entropy of a softmax policy after 2,000 iterations for each step size. The red dots correspond to runs of REINFORCE (RF), while the other colors indicate PPO with a different entropy coefficient. Each version of PPO has at least 500 individual runs.

methods in environments typically seen in RL research. In this section, we investigate PPO in our simple 2D environment and the MuJoCo Ant environment.[1] Since PPO uses a an reward bonus of $-\alpha \ln \pi(s, a, \theta)$, where a larger coefficient α encourages the policy to have more entropy, we also explore the impacts of entropy regularization in PPO on the cliff of overcommitment.

For the simple 2D environment, we repeat the experiment above and measure the number of iterations, final return, and final entropy of the policy for each step size. The results are very similar to REINFORCE, with the exception of the final entropy. So, we show the final entropy in Figure 5 and show the other measurements in Appendix E. The first thing we noticed was that PPO also exhibits the cliff of overcommitment. However, the impact of step size on final entropy is much different and depends on the entropy coefficient.

PPO has different relationship with final entropy than REINFORCE. In REINFORCE, as the step sizes increase the final entropy in the policy decreases. PPO with linear function approximation also behaves similarly. However, in PPO with neural networks, even with no entropy regularization, the entropy has 'U' shaped relationship with the step size. While the final entropy initially decreases with increasing step sizes, it eventually starts to increase. It then suddenly drops of at the cliff of overcommitment. This observation suggests there is some interesting property of the dynamics of PPO that impact entropy.

We repeat the investigation of PPO on the Ant environment, but we run the algorithm for a fixed amount of time for each step size and estimate the performance at the end of training. Additionally, we use a Gaussian policy parameterization instead of a squashed Gaussian because the action space was not bounded. So, instead of measuring entropy, we compute the logarithm of the standard deviation averaged over each action as it will reflect the spread of the actions. Figure 6 shows the results for PPO on the Ant environment with both the results at the end of optimization and during learning.

The results, generally, follow the same trend as in the other environments, e.g., the standard deviation gets smaller as the step sizes goes above the optimal value. However, at very high steps the standard deviation begins to grow larger, which, combined with the decreasing performance, likely indicates divergence. Based on these results we conclude that the overcommitment cliff is a common phenomenon across policy gradient methods.

8 Discussion and Conclusion

As mentioned before, the step size cliff does not explain all the failures in the above results, particular those with neural networks and squashed Gaussian policies. While these results make it clear that optimizing neural network policies is more challenging than linear policies, it is not clear why smaller

[1] Appendix D describes a simple modification to the Ant environment used in the experiments.

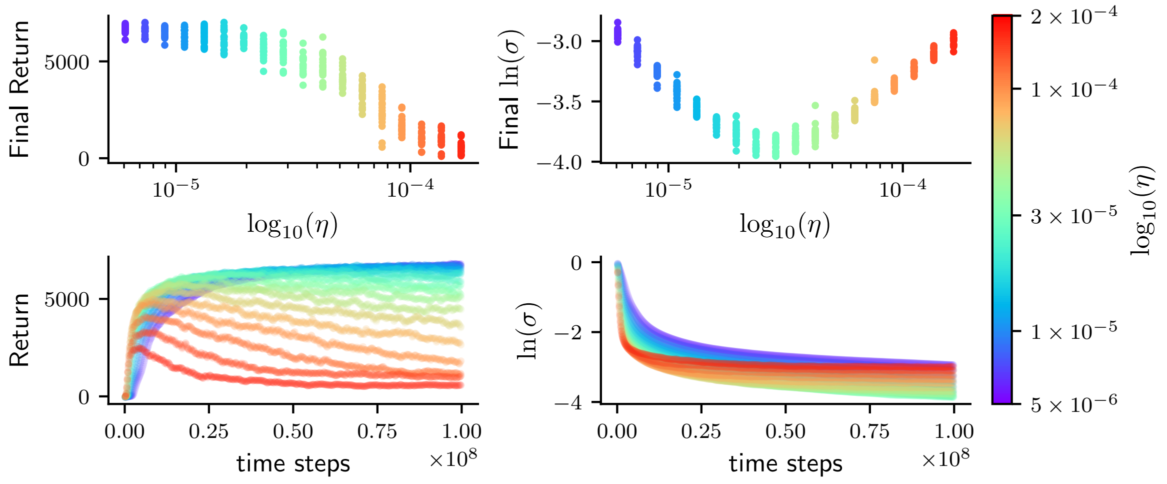

Figure 6: These plots represent the average return (left) and average log standard deviation (right) of the policy from the last 300 episodes (top row) and at each time step (bottom row) of PPO on the Ant environment. For each dot, the algorithm is run for 100,000,000 time steps. In the right plot, the log standard deviation is averaged for each action and each of the episodes. The algorithm is run 20 times for each step size. The per time step plots, average data from each run and use a windowed average to smooth the data.

step sizes were insufficient. We speculate that this has to do with with feature collapse (Lyle et al., 2022; Dohare et al., 2023), which makes it more difficult for the network to switch to another policy. For Gaussian policies, we hypothesize the extra failures are due to the policy not being able to represent actions that can go to both goals. This hypothesis is consistent with other findings that unimodal continuous action distributions can make optimization difficult (Lim et al., 2018; Sasaki & Matsubara, 2019). Nevertheless, these other failures indicate that a well tuned step sizes, alone, may be unable to have reliable convergence to a near optimal solution. There will need to be independent and compatible approaches to address each of these failures.

When designing an adaptive step size it can be tempting to think of trust region methods, which model the objective function and take a step relative to the trust of the model, as a solution. The goal of these methods is to ensure that $\rho(\theta + \eta\widehat{\nabla}) \geq \rho(\theta)$, and this approach has been presented in several forms (Pirotta et al., 2013; Papini et al., 2019; Furmston & Lever, 2015; Schulman et al., 2015; Paul et al., 2019), but our results suggest this approach is insufficient. To see why, consider that becoming greedy with respect to any action a such that $q_\pi(s, a) > v_\pi(s)$ will improve the policy, but then be suck and unable to keep exploring. A different objective is needed to ensure a better policy keeps being reached until it finds the optimal policy.

Policy gradient methods have a few special issues that making optimization difficult. The first is that the gradient can point in a direction that decreases the probability of the optimal action. This is because the value of an action can be highly dependent on the policy, e.g., the optimal action a^* can be in the sets $\arg\min_a \min_\pi q_\pi(s, a)$ and $\arg\max_a \max_\pi q_\pi(s, a)$. The second is that even if $a^* \in \arg\max_a q_\pi(s, a)$, the gradient can increase the probability of another action more than the optimal one (Schaul et al., 2019), i.e., the change in probability is proportional to $\pi(s, a)(q_\pi(s, a) - v_\pi(s))$. Natural policy gradients (Kakade, 2002) can fix this issue because they change the likelihood of each action proportional to a linear approximation of $q_\pi(s, a) - v_\pi(s)$. However, natural policy gradients require accurate estimation of $q_\pi(s, a^*)$, which is not likely if $\pi(s, a^*)$ is small. If the approximation is not sufficiently accurate then natural gradient methods are guaranteed to converge to a suboptimal policy (Chung et al., 2021; Mei et al., 2021a).

Meta learning approaches to step size adaption from supervised learning (Sutton, 1992; Mahmood et al., 2012; Degris et al., 2024) are not directly applicable to policy gradient methods. These methods can optimize the step size because in supervised learning there is a loss function that can make it clear what is or is not a too large of step, i.e., prediction error goes from negative to positive. This is not the case for policy gradients, as we do not have an objective that specifies what is too much commitment.

Several methods aim to reduce the impact of overcommitting: entropy regularization and intrinsic motivation. Entropy regularization is a technique that uses a secondary objective to encourage the policy to increase its entropy (Williams & Peng, 1991; Sabes & Jordan, 1995; Ahmed et al., 2019; Haarnoja et al., 2018). These methods can prevent complete entropy collapse, but they have additional hyperparameters, and it is unknown how to set or adapt these parameters reliably. Another technique, intrinsic motivation, applies a reward bonus to encourage the policy to explore (Agarwal et al., 2020). This last technique, in combination with restart distributions, has been shown to guarantee polynomial convergence rates for even challenging exploration environments.

In summary, based on our findings the step size has a significant impact on how much exploration the agent will perform and step sizes above the optimal value lead to overcommitment to suboptimal actions and increased training times. Thus to develop an reliable adaptive step size method for policy gradient methods, the solution will either explicitly or implicitly address the exploration exploitation trade-off.

Acknowledgments

We want to thank Martha White for feedback on an early draft of this paper.

References

Alekh Agarwal, Mikael Henaff, Sham M. Kakade, and Wen Sun. PC-PG: policy cover directed exploration for provable policy gradient learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 151–160. PMLR, 2019.

Wesley Chung, Valentin Thomas, Marlos C. Machado, and Nicolas Le Roux. Beyond variance reduction: Understanding the true impact of baselines on policy optimization. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1999–2009. PMLR, 2021.

Thomas Degris, Khurram Javed, Arsalan Sharifnassab, Yuxin Liu, and Richard S. Sutton. Step-size optimization for continual learning. *CoRR*, abs/2401.17401, 2024.

Shibhansh Dohare, Qingfeng Lan, and A Rupam Mahmood. Overcoming policy collapse in deep reinforcement learning. In *Sixteenth European Workshop on Reinforcement Learning*, 2023.

Thomas Furmston and Guy Lever. A gauss-newton method for markov decision processes. *CoRR*, abs/1507.08271, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. In *Coursera: Neural Networks for Machine Learning*. Coursera Inc., 2012.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017.

S. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, pp. 1531–1538, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Sungsu Lim, Ajin Joseph, Lei Le, Yangchen Pan, and Martha White. Actor-expert: A framework for using action-value methods in continuous action spaces. *CoRR*, abs/1810.09103, 2018.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

Ashique Rupam Mahmood, Richard S. Sutton, Thomas Degris, and Patrick M. Pilarski. Tuning-free step-size adaptation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 2121–2124. IEEE, 2012.

Jincheng Mei, Bo Dai, Chenjun Xiao, Csaba Szepesvári, and Dale Schuurmans. Understanding the effect of stochasticity in policy optimization. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems*, pp. 19339–19351, 2021a.

Jincheng Mei, Yue Gao, Bo Dai, Csaba Szepesvári, and Dale Schuurmans. Leveraging non-uniformity in first-order non-convex optimization. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7555–7564. PMLR, 2021b.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016.

Tetsuro Morimura, Eiji Uchibe, and Kenji Doya. Utilizing the natural gradient in temporal difference reinforcement learning with eligibility traces. In *International Symposium on Information Geometry and Its Applications*, pp. 256–263, 2005.

Matteo Papini, Matteo Pirotta, and Marcello Restelli. Smoothing policies and safe policy gradients. *CoRR*, abs/1905.03231, 2019.

Supratik Paul, Vitaly Kurin, and Shimon Whiteson. Fast efficient hyperparameter tuning for policy gradient methods. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Adaptive step-size for policy gradient methods. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26 (NIPS)*, pp. 1394–1402, 2013.

Philip N. Sabes and Michael I. Jordan. Reinforcement learning by probability matching. In *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, pp. 1080–1086. MIT Press, 1995.

Hikaru Sasaki and Takamitsu Matsubara. Multimodal policy search using overlapping mixtures of sparse gaussian process prior. In *International Conference on Robotics and Automation, ICRA*, pp. 2433–2439. IEEE, 2019.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *CoRR*, abs/1904.11455, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In William R. Swartout (ed.), *Proceedings of the 10th National Conference on Artificial Intelligence*, pp. 171–176. AAAI Press / The MIT Press, 1992.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Jie Tang and Pieter Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems 23*, pp. 1000–1008. Curran Associates, Inc., 2010.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Paul Wagner. A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 24, (NIPS)*, pp. 2573–2581, 2011.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

A Policy Gradient Form Equivalence

In this section, we prove that the two policy gradient forms: the policy gradient theorem and the episodic form are equivalent. We make these connections explicit but the correctness of each form has already be established and we include it here for completeness. We first start by proving a few lemmas that are helpful in proving that the forms are equivalent.

We begin by proving two lemmas regarding the episodic form.

Lemma 1 (Episode Probability)**.** *The probability of an episode τ occurring is*

$$\begin{aligned}\Pr(\tau) =& \Pr(S_0 = s_0, A_0 = a_0, R_0 = r_0, \dots, S_{T-1} = s_{T-1}, A_{T-1} = a_{T-1}, R_{T-1} = r_{T-1}) \\ &= d_0(s_0) \prod_{t=0}^{T-1} \pi(s_t, a_t, \theta) \Pr(R_t = r_t | S_t = s_t, A_t = a_t) p(s_t, a_t, s_{t+1}).\end{aligned}$$

Proof. By the Markov property

$$\Pr(A_{t+1} = a_{t+1}, R_{t+1} = r_{t+1} | S_{t+1} = s_{t+1}, S_t = s_t, A_t = a_t, R_t = r_t, \dots) = \Pr(A_{t+1} = a_{t+1}, R_{t+1} = r_{t+1} | S_{t+1} = s_{t+1})$$

and $\Pr(S_{t+1} = s_{t+1} | A_t = a_t, R_t = r_t, S_t = s_t, \dots) = \Pr(S_{t+1} = s_{t+1} | A_t = a_t, S_t = s_t)$. Thus,

$$\begin{aligned}\Pr(\tau) =& \Pr(S_0 = s_0, A_0 = a_0, R_0 = r_0, \dots, S_{T-1} = s_{T-1}, A_{T-1} = a_{T-1}, R_{T-1} = r_{T-1}, S_T = s_\infty) \\ =& \Pr(S_0 = s_0) \prod_{t=0}^{T-1} \Pr(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t) \Pr(A_t = a_t, R_t = r_t | S_t = s_t) \\ =& \Pr(S_0 = s_0) \prod_{t=0}^{T-1} \Pr(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t) \Pr(R_t = r_t | A_t = a_t, S_t = s_t) \Pr(A_t = a_t | S_t = s_t) \\ =& d_0(s_0) \prod_{t=0}^{T-1} p(s_t, a_t, s_{t+1}) \Pr(R_t = r_t | A_t = a_t, S_t = s_t) \pi(s_t, a_t, \theta) \\ =& d_0(s_0) \prod_{t=0}^{T-1} \pi(s_t, a_t, \theta) \Pr(R_t = r_t | A_t = a_t, S_t = s_t) p(s_t, a_t, s_{t+1}).\end{aligned}$$

□

Lemma 2 (Episode Derivative)**.** *The direction of steepest ascent to change policy parameters θ to make an episode more likely is*

$$\frac{\partial}{\partial \theta} \ln \Pr(\tau) = \sum_{t=0}^{T-1} \frac{\partial}{\partial \theta} \ln \pi(s_t, a_t, \theta).$$

Proof.

$$
\begin{aligned}
\frac{\partial}{\partial\theta}\ln\Pr(\tau) &= \frac{\partial}{\partial\theta}\ln\left(d_0(s_0)\prod_{t=0}^{T-1}\pi(s_t,a_t,\theta)\Pr(R_t=r_t|A_t=a_t,S_t=s_t)p(s_t,a_t,s_{t+1})\right)\\
&= \frac{\partial}{\partial\theta}\left(\ln d_0(s_0)+\ln\prod_{t=0}^{T-1}\pi(s_t,a_t,\theta)\Pr(R_t=r_t|A_t=a_t,S_t=s_t)p(s_t,a_t,s_{t+1})\right)\\
&= \frac{\partial}{\partial\theta}\left(\ln d_0(s_0)+\sum_{t=0}^{T-1}\ln\pi(s_t,a_t,\theta)+\ln\Pr(R_t=r_t|A_t=a_t,S_t=s_t)+\ln p(s_t,a_t,s_{t+1})\right)\\
&= \frac{\partial}{\partial\theta}\ln d_0(s_0)+\frac{\partial}{\partial\theta}\sum_{t=0}^{T-1}\ln\pi(s_t,a_t,\theta)+\ln\Pr(R_t=r_t|A_t=a_t,S_t=s_t)+\ln p(s_t,a_t,s_{t+1})\\
&= \underbrace{\frac{\partial}{\partial\theta}\ln d_0(s_0)}_{=0}+\sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(s_t,a_t,\theta)+\underbrace{\frac{\partial}{\partial\theta}\ln\Pr(R_t=r_t|A_t=a_t,S_t=s_t)}_{=0}+\underbrace{\frac{\partial}{\partial\theta}\ln p(s_t,a_t,s_{t+1})}_{=0}\\
&= \sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(s_t,a_t,\theta).
\end{aligned}
$$

□

The primary difference between the episodic form and the policy gradient theorem is that the episodic form uses all rewards (past and present) to weight the quality of an action. To show these forms are equivalent we need to show that the contribution of past rewards to future actions does not change the update direction of θ.

Lemma 3. *For any $t < t'$,*

$$\mathbf{E}\left[R_t\frac{\partial}{\partial\theta}\ln\pi(S_{t'},A_{t'},\theta)|S_t=s\right]=0.$$

Proof. Let $\psi(s,a) = \frac{\partial}{\partial\theta}\ln\pi(s,a,\theta)$.

$$\begin{aligned}
\mathbf{E}\left[R_t \frac{\partial}{\partial\theta}\ln\pi(S_{t'},A_{t'},\theta)|S_t = s\right] &= \sum_{r_t,s_{t'},a_{t'}} \Pr(R_t = r, S_{t'} = s_{t'}, A_{t'} = a_{t'}|S_t = s)\left(r_t\frac{\partial}{\partial\theta}\ln\pi(s_{t'},a_{t'},\theta)\right) \\
&= \sum_{r_t,s_{t'},a_{t'}} \Pr(R_t = r|S_t = s)\Pr(S_{t'} = s_{t'}, A_{t'} = a_{t'}|S_t = s) r_t\psi(s_{t'},a_{t'}) \\
&= \sum_{r_t}\Pr(R_t = r|S_t = s)\sum_{s_{t'},a_{t'}}\Pr(A_{t'} = a_{t'}|S_{t'} = s_{t'})\Pr(S_{t'}|S_t = s) r_t\psi(s_{t'},a_{t'}) \\
&= \sum_{r_t}\Pr(R_t = r|S_t = s) r_t \sum_{s_{t'}}\Pr(S_{t'}|S_t = s)\sum_{a_{t'}}\Pr(A_{t'} = a_{t'}|S_{t'} = s_{t'})\psi(s_{t'},a_{t'}) \\
&= \sum_{r_t}\Pr(R_t = r|S_t = s) r_t \sum_{s_{t'}}\Pr(S_{t'}|S_t = s)\sum_{a_{t'}}\pi(s_{t'},a_{t'},\theta)\psi(s_{t'},a_{t'}) \\
&= \sum_{r_t}\Pr(R_t = r|S_t = s) r_t \sum_{s_{t'}}\Pr(S_{t'}|S_t = s)\sum_{a_{t'}}\frac{\partial}{\partial\theta}\pi(s_{t'},a_{t'},\theta) \\
&= \sum_{r_t}\Pr(R_t = r|S_t = s) r_t \sum_{s_{t'}}\Pr(S_{t'}|S_t = s)\frac{\partial}{\partial\theta}\sum_{a_{t'}}\pi(s_{t'},a_{t'},\theta) \\
&= \sum_{r_t}\Pr(R_t = r|S_t = s) r_t \sum_{s_{t'}}\Pr(S_{t'}|S_t = s)\underbrace{\frac{\partial}{\partial\theta}1}_{=0} \\
&= 0.
\end{aligned}$$

□

Theorem 1 (Policy Gradient Form 1)**.**

$$\nabla\rho(\theta) = \sum_s\sum_{t=0}^{\infty}\gamma^t\Pr(S_t = s)\sum_a \pi(s,a,\theta)q^\pi(s,a)\frac{\partial}{\partial\theta}\ln\pi(s,a,\theta).$$

Proof. See Sutton et al. (2000). □

Theorem 2 (Policy Gradient Form 2)**.** *For any policy such that* $\frac{\partial\ln\pi(s,a,\theta)}{\partial\theta}$ *exists for all* s, a, θ, *then the gradient of* $\rho(\theta)$ *is*

$$\nabla\rho(\theta) = \mathbf{E}\left[G\sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right] = \mathbf{E}\left[\sum_{t=0}^{T-1}\gamma^t G_t\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right].$$

Proof. For the first equality:

$$\begin{aligned}
\nabla\rho(\theta) &= \frac{\partial}{\partial\theta}\rho\theta = \frac{\partial}{\partial\theta}\mathbf{E}\left[G\right]\\
&= \frac{\partial}{\partial\theta}\sum_{\tau}\Pr(\tau)G\\
&= \sum_{\tau}\frac{\partial}{\partial\theta}\left(\Pr(\tau)G\right)\\
&= \sum_{\tau}\frac{\partial}{\partial\theta}\Pr(\tau)G + \Pr(\tau)\underbrace{\frac{\partial}{\partial\theta}G}_{=0}\\
&= \sum_{\tau}G\frac{\partial}{\partial\theta}\Pr(\tau)\\
&= \sum_{\tau}\Pr(\tau)G\frac{\partial}{\partial\theta}\ln\Pr(\tau)\\
&= \sum_{\tau}\Pr(\tau)G\sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta) = \mathbf{E}\left[G\sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right].
\end{aligned}$$

For the second equality:

$$\begin{aligned}
\mathbf{E}\left[G\sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right] &= \mathbf{E}\left[\sum_{t'=0}^{T-1}\gamma^{t'}R_{t'}\sum_{t=0}^{T-1}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t'=0}^{T-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\sum_{t=0}^{T-1}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t=0}^{T-1}\sum_{t'=0}^{T-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t=0}^{T-1}\sum_{t'=0}^{t-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta) + \sum_{t=0}^{T-1}\sum_{t'=t}^{T-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t=0}^{T-1}\sum_{t'=0}^{t-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta) + \sum_{t=0}^{T-1}\sum_{t'}^{T-1}\gamma^{t}\gamma^{t'-t}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t=0}^{T-1}\sum_{t'=0}^{t-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta) + \sum_{t=0}^{T-1}\sum_{k}^{T-1-t}\gamma^{t}\gamma^{k}R_{t+k}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t=0}^{T-1}\sum_{t'=0}^{t-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta) + \sum_{t=0}^{T-1}\gamma^{t}G_t\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \underbrace{\mathbf{E}\left[\sum_{t=0}^{T-1}\sum_{t'=0}^{t-1}\gamma^{t'}R_{t'}\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]}_{=0} + \mathbf{E}\left[\sum_{t=0}^{T-1}\gamma^{t}G_t\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right]\\
&= \mathbf{E}\left[\sum_{t=0}^{T-1}\gamma^{t}G_t\frac{\partial}{\partial\theta}\ln\pi(S_t,A_t,\theta)\right].
\end{aligned}$$

□

Theorem 3 (Policy Gradient Form Equivalence)**.**

$$\nabla\rho(\theta) = \sum_s \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s) \sum_a \pi(s,a,\theta) q^\pi(s,a) \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) = \mathbf{E}\left[\sum_{t=0}^{T-1} \gamma^t G_t \frac{\partial}{\partial\theta} \ln \pi(S_t, A_t, \theta)\right].$$

Proof.

$$\begin{aligned}
\nabla\rho(\theta) =& \sum_s \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s) \sum_a \pi(s,a,\theta) q^\pi(s,a) \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) \\
=& \sum_s \sum_{t=0}^{\infty} \Pr(S_t = s) \sum_a \pi(s,a,\theta) \gamma^t q^\pi(s,a) \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) \\
=& \sum_{t=0}^{\infty} \sum_s \Pr(S_t = s) \sum_a \pi(s,a,\theta) \gamma^t q^\pi(s,a) \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) \\
=& \sum_{t=0}^{\infty} \sum_s \Pr(S_t = s) \sum_a \pi(s,a,\theta) \gamma^t q^\pi(s,a) \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) \\
=& \sum_{t=0}^{\infty} \sum_s \Pr(S_t = s) \sum_a \pi(s,a,\theta) \mathbf{E}\left[\gamma^t G_t | S_t = s, A_t = a\right] \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) \\
=& \sum_{t=0}^{\infty} \sum_s \Pr(S_t = s) \sum_a \pi(s,a,\theta) \mathbf{E}\left[\gamma^t G_t \frac{\partial}{\partial\theta} \ln \pi(s,a,\theta) | S_t = s, A_t = a\right] \\
=& \sum_{t=0}^{\infty} \sum_s \Pr(S_t = s) \mathbf{E}\left[\gamma^t G_t \frac{\partial}{\partial\theta} \ln \pi(s, A_t, \theta) | S_t = s\right] \\
=& \sum_{t=0}^{\infty} \mathbf{E}\left[\gamma^t G_t \frac{\partial}{\partial\theta} \ln \pi(S_t, A_t, \theta)\right] \\
=& \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t G_t \frac{\partial}{\partial\theta} \ln \pi(S_t, A_t, \theta)\right] \\
=& \mathbf{E}\left[\sum_{t=0}^{T-1} \gamma^t G_t \frac{\partial}{\partial\theta} \ln \pi(S_t, A_t, \theta)\right].
\end{aligned}$$

□

B Hyperparameter Details:

This section lists the hyperparameters used for each experiment.

For the 2D environment we used a tile coding basis function with 16 tilings of 4 tiles per dim and a two layer neural network with 32 hidden units and relu activations. For each iteration of REINFORCE 50 episodes were used to estimate the gradient. PPO also collected 50 episodes per iteration, performed 10 policy updates in mini-batches, a clip ratio of 0.2, and $\lambda = 1$. We also do not use a version of PPO with a KL divergence penalty. Both algorithms use $\gamma = 1$.

For the Ant environment PPO used a neural network with two hidden layers of size 64. It performed updates after every 3072 time steps, using mini-batches of 32, and went over the data 10 times per batch. The clipping parameter was set to 0.1, $\gamma = 0.98$, $\lambda = 0.8$, and no entropy bonus.

C Entropy Computation

For the continuous actions, using differential entropy can lead to negative or very large values and not necessarily reflect the measure of uncertainty we want to capture. What we want to measure is how much the action distribution covers the space $[-1, 1]$. So we discretize the action space into 1,000 equal width bins. We then create a discrete probability distribution based on the total probability that an action is sampled in each bin. For example for a bin $u_i = [a_i, a_{i+1})$ with a_i and a_{i+1} being the lower and upper bounds of the bin, the probability of the selected bin U being u_i in state s is

$$\Pr(U = u_i) = \int_{a_i}^{a_{i+1}} \pi(s, a, \theta) da = F_A(a_{i+1}) - F_A(a_i),$$

where F_A is the cumulative distribution function of the action A as defined by π. We compute the entropy of U to represent the spread of the action distribution.

This discretization has maintains the properties we care about, e.g., the entropy is maximized when the distribution of A is uniform and the entropy goes towards 0 as the distribution becomes deterministic. Furthermore, it is a better measure of spread than standard deviation for the squashed Gaussian. When the standard deviation becomes large more probability mass gets put on the end points $-1, +1$ than in the middle thus there would not many substantially different actions being tried even though the standard deviation would be large. In this case the entropy discretized distribution would be relatively small since the probability mass is mostly on two bins.

D Modification to the Ant environment

The Mujoco Ant environments (all versions) define an is_healthy status for the simulated robot that terminates the episode if the Ant is unhealthy. Specifically, if the ant is too low to or too far from the ground, the episode ends. However, the Ant can get stuck on its back; this is not always considered unhealthy, and the result is that a great deal of compute may be spent on simulating the Ant stuck upside-down. Since this is clearly an oversight, we use a modification to the Ant environment that modifies the is_healthy logic to account for this situation. That is, the episode ends if the Ant flips upside-down.

E Additional Results

This section shows additional resuts. Figures 7 and 8 show the measurements for each step size on the 2D environment for both softmax and squashed Gaussian distributions. Figures 9 and 10 show the measurements for REINFORCE using the Adam optimizers. Figure 11 shows the measurements for PPO with a softmax distribution on the same environment.

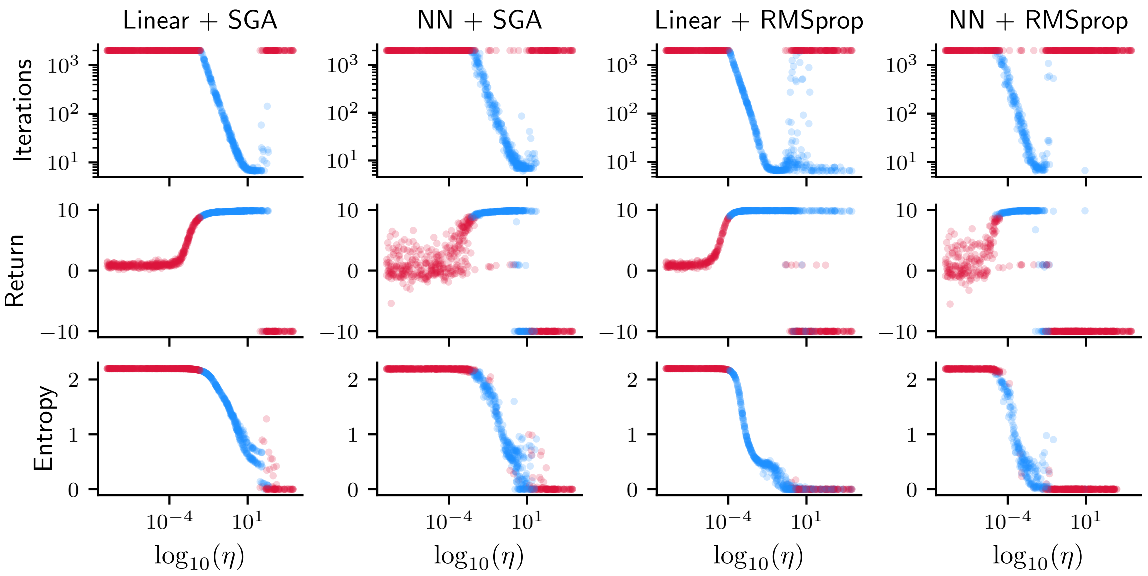

Figure 7: This figure show the results of REINFORCE using the softmax parameterization.

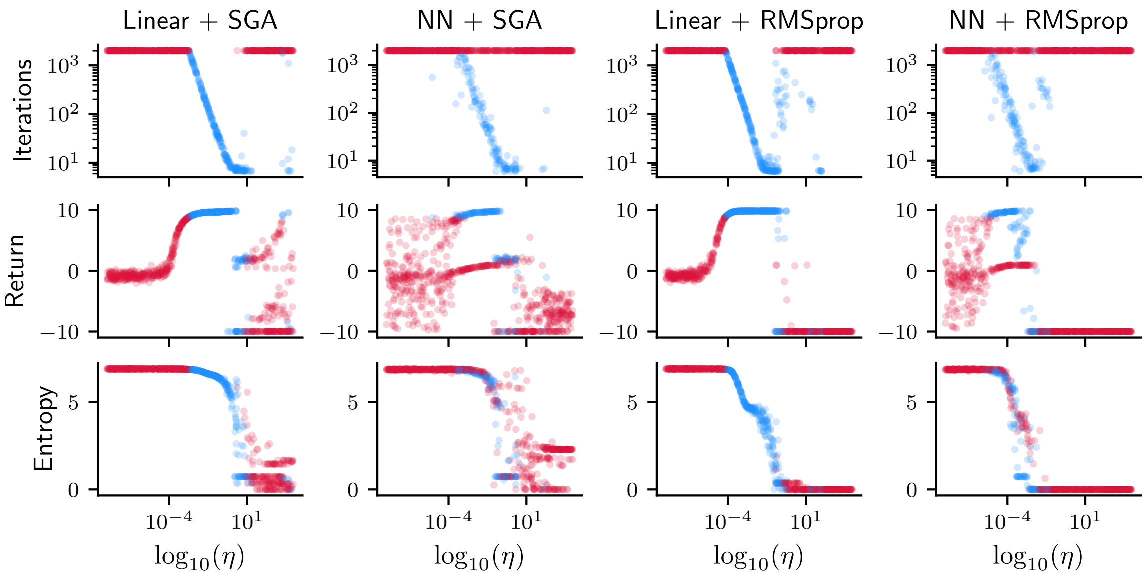

Figure 8: This figure show the results of REINFORCE using the squashed Gaussian parameterization.

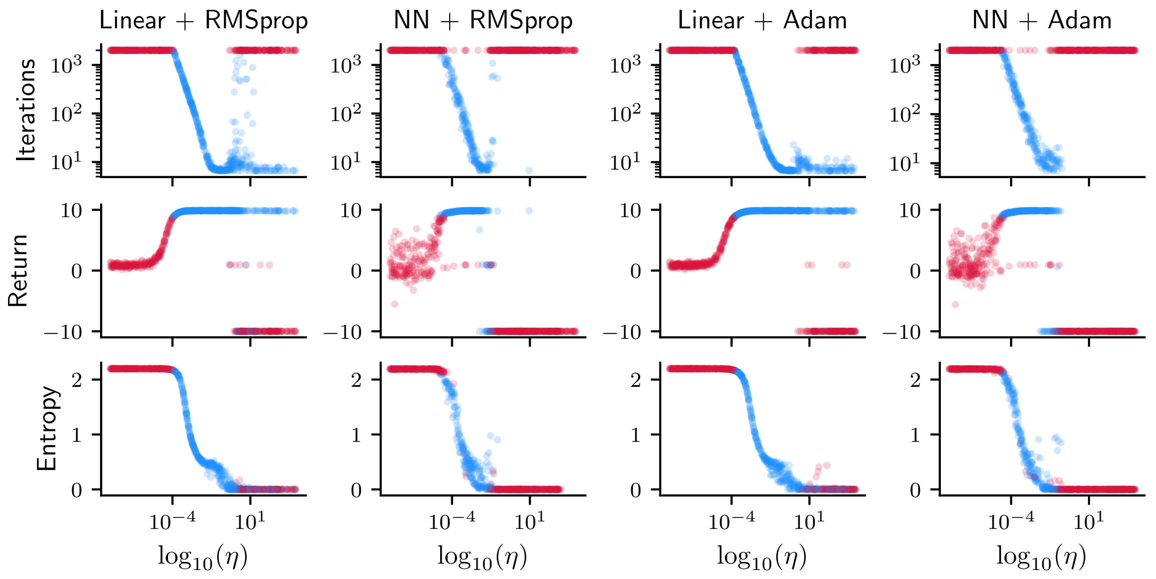

Figure 9: This figure compares the results of REINFORCE using the softmax parameterization with RMSprop and Adam. Notice that the shape of all the plots for the two optimizers are very similar.

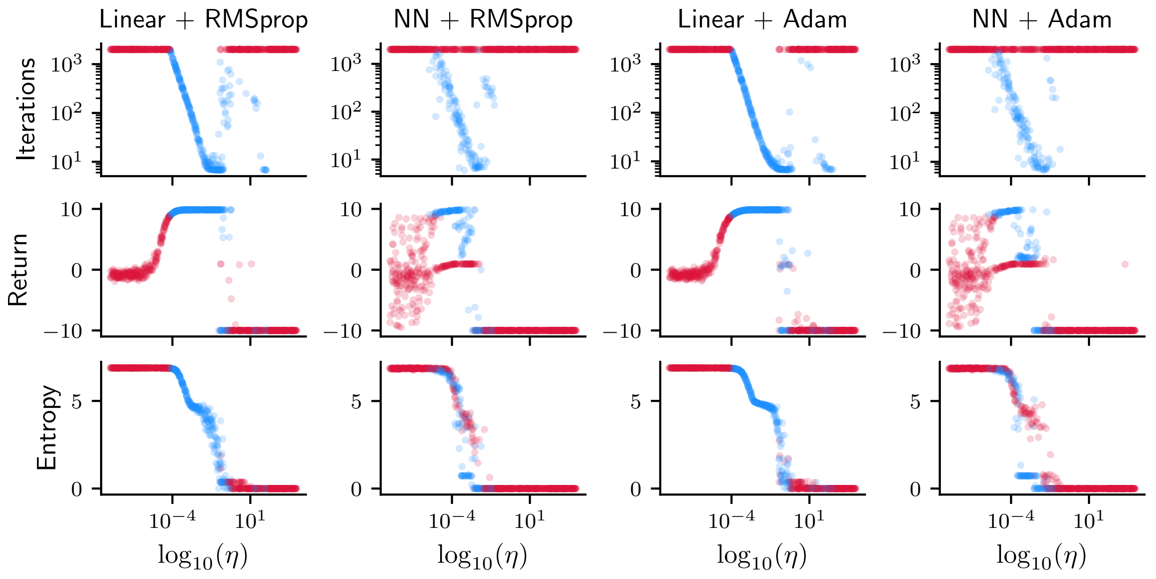

Figure 10: This figure compares the results of REINFORCE using the squashed Gaussian parameterization with RMSprop and Adam.

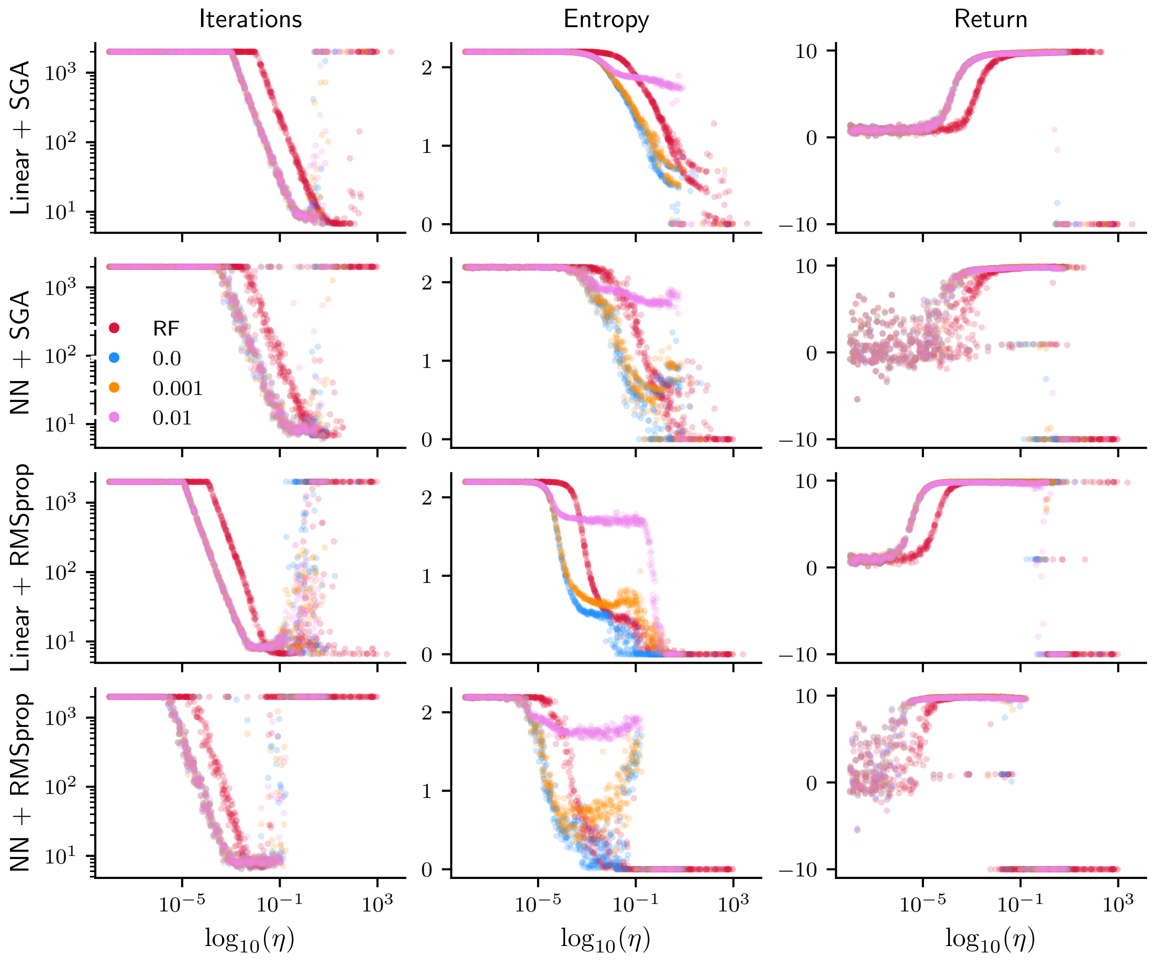

Figure 11: This plot shows the measurements comparing PPO with softmax parameterization to REINFORCE on the 2D environment. The red dots correspond to REINFORCE, while the other colors indicate using a specific entropy coefficient with PPO.