On-Policy Maximum Entropy Deep Reinforcement Learning

1 Introduction

Model-free deep reinforcement learning (RL) algorithms have succeeded in a variety of tasks, including games [MKS+13], robotics [SLA+15], and traffic control [TPSS19]. These algorithms have been successful in such complex domains due to their application of high-capacity function approximators. These algorithms fall into two families: *on-policy* algorithms (e.g. TRPO [SLA+15], PPO [SWD+17], A3C [MBM+16]) and *off-policy* algorithms (e.g. Q-Learning based methods [MKS+15], DDPG [LHP+15]). Whereas off-policy algorithms may reuse previous experiences, on-policy algorithms require new samples to be collected for each gradient step. However, despite the increased sample complexity, on-policy algorithms have become the tool-of-choice for many continuous control problems. Indeed, PPO was shown to outperform Rainbow [HMVH+18], a combination of Q-learning approaches, in most of the levels in the ProcGen benchmark (including the FruitBot level) [CHHS20]. Although successful in these domains, deep RL algorithms can be difficult to train due to stability and convergence issues [MSB+09]. This problem is exacerbated in certain environments, such as those with sparse rewards [TZXS19].

One popular approach to combat this problem is to encourage agents to seek policies have high *entropy*. Intuitively, by optimizing for policies that have higher entropy, the agent's actions will tend not to spiral into a purely exploitative or deterministic pattern. This observation lead to the ad-hoc addition of "entropy regularization" terms to the objectives of state-of-the-art deep RL agents [MBM+16, SLA+15, SWD+17]. However, further theoretical analysis into the effectiveness of these heuristic regularizers led to a richer theory maximum entropy reinforcement learning that gives a probabilistic graphical model interpretation for the reinforcement learning problem [Lev18]. From this interpretation, a more principled approach to introducing entropy into reinforcement learning may be derived: the maximum entropy objective. The exact difference between entropy regularization and the maximum entropy objective is nuanced – we describe this in more detail in the next section. Although entropy regularization is common in on-policy algorithms such as PPO, the more principled maximum-entropy objective has only been well-studied in off-policy algorithms [HTAL17, HZAL18, SSW19]. We aim to improve the performance of the PPO algorithm by *replacing* entropy regularization with a maximum entropy objective. We consider training and test reward, in addition to the speed of training convergence, as metrics of agent performance, and find that entropy maximization achieves superior performance in all three categories with fewer time steps and less training data.

2 Related Work: Maximum Entropy vs Entropy Regularization

Maximum Entropy Objectives. In maximum entropy reinforcement learning [ZMBD08, RTV12], the agent aims to optimize the expected reward in addition to the expected entropy of the policy. Formally, given a horizon T, we seek to maximize the objective

$$\sum_{t=1}^{T} \underbrace{\mathrm{E}\left[r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]}_{\text{Expected reward}} + \beta \underbrace{\mathrm{E}\left[-\log \pi(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]}_{\text{Policy entropy}}$$

where r denotes the reward function and π the policy. The parameter β is used to calibrate the entropy term, and it is often tuned or annealed in practice. This augmentation of the reward makes policies more robust to model and estimation errors, and further increases the propensity for agents to acquire more diverse behaviors [HTAL17]. Further, the maximum entropy objective may be derived in a principled way from an interpretation of reinforcement learning as probablistic inference [Lev18]. In this interpretation, states are deemed "optimal" with probability exponential in the reward signal, and the agent aims to maximize the probability of landing in these states. Such probabilistic optimization leads to "soft" versions of classical Bellman value and Q-functions

$$V(\boldsymbol{s}_t) = \log \int_{\mathcal{A}} \exp\left(Q(\boldsymbol{s}_t, \boldsymbol{a}_t)\right) d\boldsymbol{a}_t \approx \max_{\boldsymbol{a}_t} Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$$

This insight has led to "soft" analogues of standard off-policy deep reinforcement learning algorithms, including Soft Q-Learning [HTAL17], Soft Actor-Critic [HZAL18], and Soft DDPG [SSW19]. To the best of our knowledge, we are not aware of any similar principled maximum entropy analogues of on-policy algorithms (including PPO). These algorithms instead tend to use *entropy regularization*, a technique we describe below.

Entropy Regularization. Another method of encouraging agents to seek high entropy policies is a heuristic approach known as entropy regularization, and is standard in state-of-the-art on-policy algorithms [MBM+16, SWD+17,

OMKM16]. This regularization takes place either by explicitly adding an entropy term to the policy gradient,

$$\boldsymbol{\theta} - \boldsymbol{\theta}_{\texttt{old}} \propto \mathrm{E}_{s,a}\left[Q^{\pi}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi(s,a)\right] + \beta \underbrace{\mathrm{E}_{\boldsymbol{s}}\left[\nabla_{\boldsymbol{\theta}} \mathrm{E}_{\boldsymbol{a}}\left[-\log \pi(\boldsymbol{a} \mid \boldsymbol{s})\right]\right]}_{\text{Entropy bonus}}$$

such as in [OMKM16], or by adding an entropy term to the loss function as in the PPO algorithm [SWD+17] (we describe this latter form of regularization in the next section).

Although entropy regularization has a similar effect to the use of a maximum entropy objective, the two approaches are not equivalent. The following explanation, due to Levine, makes this distinction more clear:

> "Entropy regularization is not, in general, equivalent to the maximum entropy objective, which not only optimizes for a policy with maximum entropy, but also optimizes the policy itself to visit states where it has high entropy. Put another way, the maximum entropy objective optimizes the expectation of the entropy with respect to the policy's state distribution, while entropy regularization only optimizes the policy entropy at the states that are visited, without actually trying to modify the policy itself to visit high entropy states." ([Lev18], Section 5.2)

3 Background: Deriving the Maximum Entropy PPO Objective

In this section, we introduce the PPO-clipped objective function that we use as a baseline for our experiments, and we derive a novel modification to the objective for a maximum entropy agent. We consider an actor and critic with shared parameters $\boldsymbol{\theta}$.

The PPO Objective. In the PPO algorithm, the total objective (actor objective + critic objective) is regularized by an entropy bonus term:

$$L_t^{REG}(\boldsymbol{\theta}) = \underbrace{\hat{\mathrm{E}}_t\left[\min\left(\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}\hat{A}_t,\ \texttt{clip}\left(\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]}_{\text{PPO-Clip objective}} - \alpha \underbrace{\hat{\mathrm{E}}_t\left[\left(V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) - \hat{R}_t\right)^2\right]}_{\text{Value Function loss}}$$
$$+ \beta \underbrace{\hat{\mathrm{E}}_t\left[\mathrm{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}_t)}\left[-\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}_t)\right]\right]}_{\text{Entropy bonus}}$$

Here, $\hat{E}_t$ denotes an empirical expectation over a minibatch of examples, $\hat{A}_t$ denotes the λ-generalized advantage estimate,

$$\hat{A}_t = \sum_{i=t}^{T-1} (\gamma\lambda)^{i-t}\left(r(\boldsymbol{s}_i, \boldsymbol{a}_i) + \gamma V_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{s}_{i+1}) - V_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{s}_i)\right)$$

and $\hat{R}_t = \hat{A}_t + V_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{s}_t)$ denotes the (discounted) return estimate. Each training iteration, the objective is maximized with gradient descent. New sample trajectories must be regularly collected as this is an on-policy algorithm.

A Maximum-Entropy Implementation. We make the following novel modification to the PPO loss function that corresponds to a proper maximum entropy objective

$$L_t^{MAX}(\boldsymbol{\theta}) = \hat{\mathrm{E}}_t\left[\min\left(\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}\hat{A}_t(\beta),\ \texttt{clip}\left(\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t(\beta)\right)\right] - \alpha\hat{\mathrm{E}}_t\left[\left(V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) - \hat{R}_t(\beta)\right)^2\right]$$

where we modify the advantage estimate $\hat{A}_t(\beta)$ and the estimated return $\hat{R}_t(\beta) = \hat{A}_t(\beta) + V_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{s}_t)$ to incorporate the entropy term

$$\hat{A}_t(\beta) = \sum_{i=t}^{T-1} (\gamma\lambda)^{i-t}\left(\underbrace{r(\boldsymbol{s}_i, \boldsymbol{a}_i) + \beta \mathrm{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}_i)}\left[-\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}_i)\right]}_{\text{Maximum entropy objective}} + \gamma V_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{s}_{i+1}) - V_{\boldsymbol{\theta}_{\texttt{old}}}(\boldsymbol{s}_i)\right)$$

4 Methods/Approach: Handling Complexity

Modifying off-policy algorithms to consider maximum entropy objectives is highly tractable: one need only subtract $\log \pi(\boldsymbol{a} \mid \boldsymbol{s})$ from the reward of the current timestep. Unfortunately, for on-policy methods like PPO that make use of advantage estimates, this process becomes far more complex. Observe that our modified expressions for the advantage and reward estimates $\hat{A}_t(\beta)$ and $\hat{R}_t(\beta)$ make use of the model parameters $\boldsymbol{\theta}$. Thus, in the backward pass of our modified objective L_t^{MAX}, gradients have to flow through these estimates – each of which is a sum of $O(T)$ terms. In practice, this is highly intractable: when we attempted to implement this, runtime increased from 20-40 minutes to 13-20 hours and GPU usage from 2 GB to 8 GB. To make this more tractable, we used the following (one-step) approximation for the advantage estimate, which requires only one gradient computation in the backward pass:

$$\begin{aligned}\hat{A}_t(\beta) &= \sum_{i=t}^{T-1} (\gamma\lambda)^{i-t} \left(r(\boldsymbol{s}_i, \boldsymbol{a}_i) + \beta \mathrm{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}_i)} \left[-\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}_i) \right] + \gamma V_{\boldsymbol{\theta}_{\mathtt{old}}}(\boldsymbol{s}_{i+1}) - V_{\boldsymbol{\theta}_{\mathtt{old}}}(\boldsymbol{s}_i) \right) \\ &\approx \underbrace{r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \beta \mathrm{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}_t)} \left[-\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}_t) \right] + \gamma V_{\boldsymbol{\theta}_{\mathtt{old}}}(\boldsymbol{s}_{t+1}) - V_{\boldsymbol{\theta}_{\mathtt{old}}}(\boldsymbol{s}_t)}_{\text{First step advantage estimate with entropy}} + \underbrace{(\gamma\lambda)\hat{A}_{t+1}(0)}_{\text{Unaltered advantage estimate}}\end{aligned}$$

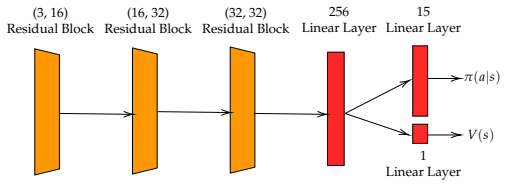

Figure 1: **Model architecture.** We used the IMPALA [ESM+18] architecture to learn an actor and critic that use shared features.

Using the equation $\hat{R}_t(\beta) = \hat{A}_t(\beta) + V_{\boldsymbol{\theta}_{\mathtt{old}}}(\boldsymbol{s}_t)$, this approximation gives a one-step approximation for the returns as well. We emphasize that we are not aware of any instances of our modified objective L_t^{MAX} in the literature. As such, this one-step approximation is also of our own design. Our approximation is, however, motivated by similar one-step approximations in the policy gradient entropy regularization literature (see e.g. Section 6.1 of [SCA18]). To implement our approach, we modified the OpenAI baseline implementation of PPO [DHK+17], using IMPALA [ESM+18] to learn an actor and critic with shared image features. Figure 1 gives a graphical depiction of our architecture. We compare the original entropy-regularized L_t^{REG} objective with the one-step approximation to our L_t^{MAX} objective.

5 Experimental Design and Results

We train three types of agents on the FruitBot level in `easy_mode` on $\{50, 100, 250, 500\}$ training levels. The first agent is trained using the PPO algorithm with no entropy augmentations. The second is trained with entropy regularization, using the L_t^{REG} objective as given in [SWD+17, CHHS20]. The third agent uses the PPO algorithm under our one-step approximation of our maximum entropy objective L_t^{MAX}. The extent to which the entropy affects the objective for the latter two agents is controlled by a temperature coefficient, which can be further adjusted using an exponential annealing parameter. We tune each agent's temperature, keeping other parameters as in [CHHS20], then benchmark their performance across training and additional unseen levels.

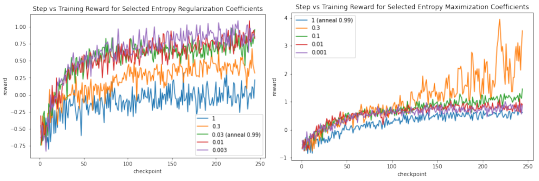

Figure 2: Training rewards for selected tuning runs of the regularization parameter (left) and maximization parameter (right) All runs were conducted with 100 training levels for 2000000 steps on `hard_mode`, to provide for greater differentiation.

We evaluated our agents based on three primary factors: the speed of convergence, the reward at convergence, and the entropy achieved by each agent's learned policy. These three metrics formed the core of our decision-making process and guided our experimentation forward. We evaluated the speed of convergence based on graphs of training reward for each agent, considering the number of training steps it took before each crossed certain mean reward thresholds on previously seen levels. The converged reward for each agent, conversely, was evaluated based on performance on unseen levels after training had been completed. This allowed us to understand whether our agents were overfitting to their known distributions at the expense of generalization, which we hoped entropy maximization would help prevent. Finally, we created graphs of the average policy entropy for each agent over the course of training, which helped us visualize whether we were achieving our goal of learning a diverse distribution over actions.

Tuning runs were performed using a logarithmic grid search across entropy parameters between 0.0001 and 1, with different runs roughly a factor of 3 apart. Agents were trained for each entropy parameter and method on the same settings and seed. The best performing orders of magnitude were then further explored, with finer-grained search and the addition of annealing parameters. The results for tuning on both one-step entropy maximization and entropy regularization are shown in Figure 2. The optimal tuned entropy values were 0.01 and 0.3 for entropy regularization and one-step entropy maximization, respectively. All further comparison runs were performed with these optimal entropy coefficients, to provide a fair perspective on the best performance of each method.

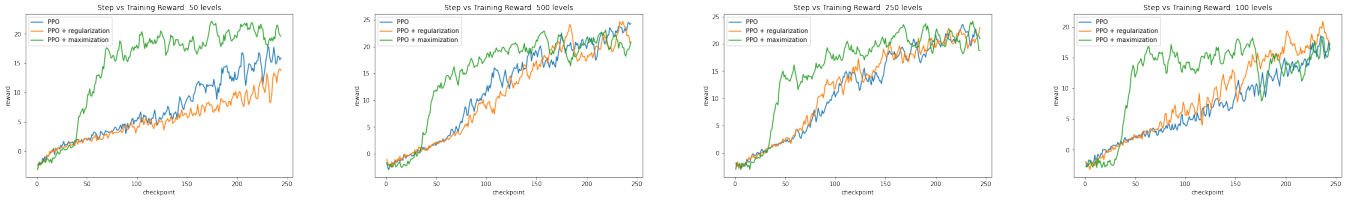

Figure 3: Training rewards for a PPO agent with no entropy regularization/maximum entropy objective (in blue), entropy regularization (in orange), and our maximum entropy objective (in green) with identical seeds and entropy temperature parameters set to the optimal value as determined above.

The plots in Figure 3 show the reward during training for each agent; notice that our entropy maximization agent tends to reach higher levels of performance across all four level counts significantly earlier in the training process than either of the other agents. Crucially, this is also reflected when these models are tested on unseen data (Figure 4). At timestep 1,000,000 (halfway through training), our agent achieved higher test rewards than the previous models, which demonstrates that our model was not simply overfitting to achieve the observed results.

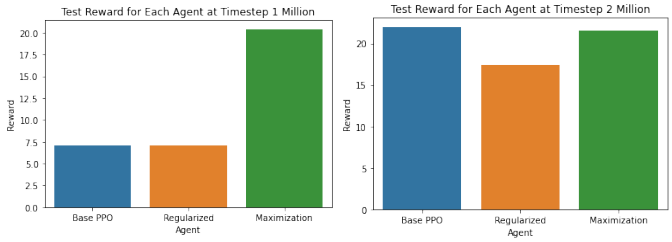

Figure 4: Test results for our models at 1 million timesteps (far left) and 2 million timesteps (center). These figures were generated using models trained on the first 500 levels in `easy_mode`, and tested on levels 500 through 1000 of ProcGen FruitBot. Notably, **our agent achieves test rewards of around 20** far earlier than either alternative.

Interestingly, Figure 5 shows that the entropy maximization agent actually had *lower* policy entropy over time. This drop in entropy occurs roughly at the same time when the agent experiences a spike in training reward. This indicates that the entropy maximizing agent was able to quickly identify certain behaviors that increased the scale of the reward signal, and correspondingly adopted a more deterministic policy that exploits these behaviors. We also experimented with a combined agent that uses entropy regularization in addition to a maximum entropy objective. This agent (shown in red) performs poorly, and seems to only maximize entropy. It is likely that this poor performance is due in part to difficulties in hyperparameter tuning, as the interaction of these two terms is far more complicated and difficult to optimize than either alone.

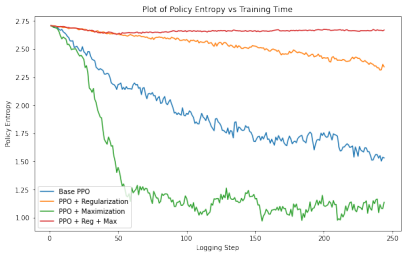

Figure 5: Policy entropy over time for selected training runs (250 levels) showing clear differentiation in entropy levels across training methods.

6 Conclusion

In this paper, we modified the PPO-Clipped objective to tractably approximate a maximum entropy objective. Our experimental results indicate that PPO under this modified objective yielded better performing and more robust results in fewer timesteps and using less training data, when compared to the state-of-the-art entropy-regularized objective. We see our results as a nuanced confirmation of the well-known success of entropy augmentations to reinforcement learning objectives. Current state-of-the-art on-policy and off-policy algorithms use entropy in some fashion. Our results demonstrate that not all entropy augmentations are equal – the more principled maximum entropy approach is superior to heuristic entropy regularization approaches in practice. Further, our one-step approximation to the maximum entropy objective is as easy and tractable to implement as the existing entropy-regularized objective. Thus, we believe that our approach motivates the further use of proper entropy maximization, and can be used to improve the performance of other popular on-policy algorithms (e.g. TRPO, A3C).

7 Contributions

Naveen (Related Work, Derivations, Research Direction): 33%; Will (Experimental Design, Tuning, Logging): 33%; Kamyar (Code, Environment Setup, Methods/Approach): 33%; CS 182 Overlords 1%

References

[CHHS20] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[DHK+17] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[ESM+18] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 10–15 Jul 2018.

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[HTAL17] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[HZAL18] 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.

[Lev18] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[LHP+15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[MBM+16] 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.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[MSB+09] Hamid Reza Maei, Csaba Szepesvari, Shalabh Bhatnagar, Doina Precup, David Silver, and Richard S Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In *NIPS*, pages 1204–1212, 2009.

[OMKM16] Brendan O'Donoghue, Remi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

[RTV12] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

[SCA18] John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning, 2018.

[SLA+15] 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.

[SSW19] Wenjie Shi, Shiji Song, and Cheng Wu. Soft policy gradient method for maximum entropy deep reinforcement learning. *arXiv preprint arXiv:1909.03198*, 2019.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[TPSS19] Kai Liang Tan, Subhadipto Poddar, Soumik Sarkar, and Anuj Sharma. Deep reinforcement learning for adaptive traffic signal control. In *Dynamic Systems and Control Conference*, volume 59162, page V003T18A006. American Society of Mechanical Engineers, 2019.

[TZXS19] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *arXiv preprint arXiv:1911.01417*, 2019.

[ZMBD08] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.