Comparing model-free reinforcement learning algorithms

Kristjan Reba
kr3377@student.uni-lj.si

The Faculty of Computer and Information Science, University of Ljubljana

***Abstract*—In recent years there was a lot of progress made on hard to solve tasks such as vision and continuous control which is essential in robotics. Most of this progress is due to increase in computing power, large amounts of data and better algorithms. Reinforcement learning (RL) has been able to achieve good results on difficult tasks such as the game of Go and Atari 2600 benchmark. In this paper we compare 3 different reinforcement learning algorithms on 3 different tasks. Algorithms considered are Asynchronous Actor Critic (A2C), Proximal Policy Optimization (PPO) and Soft Actor Critic (SAC). We find out that SAC outperforms other two algorithms in terms of final reward achieved, while also being more robust and faster to converge.**

***Index Terms*—neural network, agent, continuous control**

I. Introduction

Reinforcement learning (RL) has in recent years shown incremental progress on continuous control tasks. The promise of such algorithms is solving previously hard to solve problems such as robotic control and hard optimization problems [12]. With large amount of computing power and quality simulated environments for the agents to learn on, large progress has been made in improving the RL algorithms [3], [25]. RL algorithms augmented with neural networks have been able to solve previously unsolved control tasks such as the Atari 2600 benchmark, game of Go and StarCraft that are hard even for humans and are considered intractable for brute force algorithms [15], [24], [30]. Currently the best RL agents are based on model-free reinforcement learning, which means the agents make decisions without having an internal model of the environment [25]. Meaning they don't keep track of transition function and reward function. The agents that keep track of those two functions are called model-based algorithms. Even though model-free algorithms are able to perform well on simulated environments they have hard time transitioning to real world scenarios. Many of the reinforcement learning benchmarks do not capture the richness of the real world needed for robust multipurpose agents that would help solve real world problems. The promise of simulated environments used as a benchmark is fast and standardized comparison of algorithms and their performance. There is a high correlation of performance in simulated environment in relation to real world one. The reason real world tasks are so difficult for reinforcement learning agent is the infinite continuous action space, limited state information that is described by an observation by noisy sensors. Agent performing tasks in physical environment are also limited by the time it takes to perform an action and the time it takes to make an observation. Having low sample complexity and fast convergence speed make algorithms more suitable for learning in real world. In simulation we don't have to worry about the speed of actuators and sensors which makes it a suitable training environment.

In this paper we are interested in comparing model-free reinforcement learning algorithms on a couple of simulated tasks from OpenAI Gym [3]. We are empirically evaluating three model-free reinforcement learning algorithms on three different tasks. The algorithms: Asynchronous Actor Critic (A2C) [14], Proximal policy optimization (PPO) [23] and Soft Actor Critic (SAC) [9] are all currently state of the art algorithms used for solving complex continuous control tasks and are suitable for training in simulated environments. Our goal throughout this paper is to introduce the ideas behind the reinforcement learning paradigm and show state of the art algorithms performing on tasks that would have only a couple of years ago been unsolvable by computer in manageable amount of time.

II. Related work

In this chapter we briefly describe the ideas presented to the field of reinforcement learning in the recent years.

The idea of policy gradient algorithms is taking steps on a policy that improve the agents performance. Policy is usually represented by neural network and optimization step is usually a single step of gradient descent adjusting the weights of the network. The goal is to suppress the actions that lead to low reward and increase the probability of actions that lead to high rewards all while taking into consideration the state the agent is in [25], [26].

In [22] authors present an algorithm called Trust Region Policy Optimization (TRPO) that updates policies by taking the largest step possible. The step taken by TRPO is constrained by the KL-divergence of the new policy in relation to the new one. This prevents the algorithm from taking too large of a step and puts a constraint on how close the policies have to be. This approach is closely connected to Proximal Policy Optimization (PPO) described in the next section.

We can split the RL algorithms in two distinct groups. Agent can either have access to the model of the environment, meaning it has a transition function that predicts state transitions and subsequent rewards, or the agent has access to

no such function. As presented in [24] an agent can be given a model of an environment to achieve superhuman performance on complex tasks. The promise of model-based algorithms is using the transition function to plan ahead and evaluate many different action choices for a given state. On benefit of this type of algorithms is reduced sample complexity, but it comes at a cost of providing an accurate model of the environment. Recently it has been demonstrated that learning a model of the environment while learning to achieve high rewards, can achieve even better results than when the model is given [21]. Model-free methods are known for their sample inefficiency, meaning they need many more samples to learn a complex task compared to model-based algorithms. Even though model-free algorithms are less sample efficient they are much easier to implement and tune.

There have been many more attempts to make the agents perform better on complex tasks. One useful augmentation of algorithm is using episodic memory as described in [2] and [20]. For making good policy improvement we can either rely on just the recent data sample (observation and reward received) or we can take a step on the policy based on a history of actions, observations and rewards. We can even control which data samples to store in memory based on some measure of importance or novelty. One increasingly popular approach to reinforcement learning is meta-learning, where we try to learn a good reinforcement learning algorithm from many different tasks as described in [6], [13], [7]. Meta-learning agent uses a distribution of tasks to train on and if trained well, an agent can learn very fast on some yet unseen task. A hope with meta-learning algorithm is to someday create a general enough algorithm that can quickly learn any new task we give it to solve. We could also try to learn simpler tasks and then try to use those for completing the more challenging ones that are composed of simpler tasks, such as first learning to walk and then navigating the maze to reach the right goal. This approach is called hierarchical learning and is described in [28], [16], [29]. For the agent to efficiently learn (reduce sample complexity) it has to optimize its policy in big steps with each data sample. But low quality of the data sample can be disastrous for an agent. Because the data samples (observations) depend on agent acting on the environment, an agent can get stuck in a couple of environment states without leaning anything outside of this states. The problem of bad data samples can be solved by better exploration strategies [5], [4] that make sure agent gets as much diverse experiments from the environment as possible and aim to prevent an agent from getting stuck in bad policy without learning anything new [18], [8], [27]. In order to increase sample complexity of reinforcement learning algorithms and to incorporate some human knowledge about the task there has been many algorithms proposed for imitation learning [17], [11]. In imitation learning an agent has access to human demonstration performing the task and uses this information to improve its own policy to match the assumed human policy [19].

III. Methodology

We consider a standard reinforcement learning setting in the form of infinite-horizon Markov decision process (MDP). To use the formalism of Markov decision process we must assume that the system obeys the Markov property: transition to the next state only depends on the current state and the action taken and not on the prior state history.

In reinforcement learning setting an MDP is described by a tuple (S, A, R, P, p_0) where:

- S is a set of all states in which the agent can find it's self in.
- A is a set of all actions an agent can take.
- $R : S \times A \times S \to R$ is a reward function where $r_t = R(s_t, a_t, s_{t+1})$
- $P : S \times A \to P(S)$ is a transition probability function where $P(s'|s, a)$ describes the probability of transitioning to state s', if an agent is in state s and takes action a.
- p_0 describes the starting state distribution.

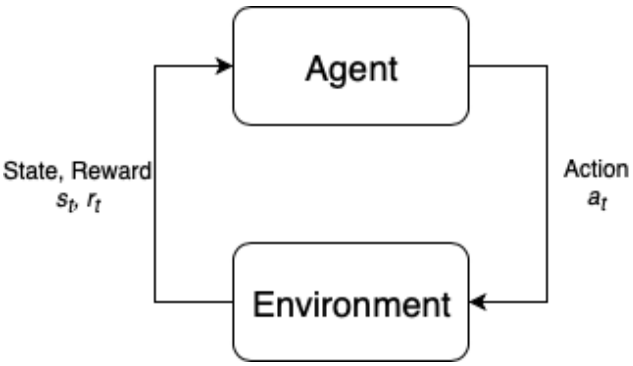

Fig. 1. Reinforcement learning loop, where agent acts on the environment with action a_t and receives a state and a reward from the environment on each time-step t.

A. Asynchronous Actor Critic

Asynchronous Actor Critic (A2C) is an algorithm which can be viewed as 2 separate entities (actor and critic) controlling the agent [14]. The actor takes as input the state and then outputs the best action. It essentially controls how the agent behaves by learning the optimal policy (policy-based). The critic, on the other hand, evaluates the action by computing the value function (value based). Those two models participate in a game where they both get better in their own role as the time passes. The result is that the overall architecture will learn to play the game more efficiently than the two methods separately.

B. Proximal Policy Optimization

Proximal Policy Optimization (PPO) is similar to TRPO and is trying to solve the problem of the biggest step we can take on the policy using the data we currently have, without stepping too far and causing performance to worsen [23]. PPO methods are significantly simpler and empirically perform just es well as TRPO. For the PPO to match the performance of TRPO the authors got rid of KL-divergence measure for

the policy update and instead rely on policy clipping. Policy clipping makes sure that the updated policy is in some sense close to the old policy [23].

C. Soft Actor Critic

Soft Actor Critic (SAC) is an algorithm that optimizes a stochastic policy [9]. If an agent is optimizing the stochastic policy that means that the agent can explore the state space without always taking the same action. The output of the agent is a probability distribution of actions for a given state. The central feature of SAC is entropy regularization. The agent regulates how much entropy (rough measure of randomness) there is in its policy meaning that the action probability distribution has the highest entropy possible. This is a simple way to approach the exploration-exploitation dilemma that has been in the center of reinforcement learning for a long time. If we change the policy so that the entropy increases we make the agent explore more while reducing entropy means we make the agent perform more or less the same actions in each state. With increased entropy the agent can accelerate learning later on while also preventing premature convergence to a bad local optimum.

In the next subsections we introduce the environments we use evaluate the presented algorithms.

D. Task 1: Mountain Car

We evaluate the agents on 3 different continuous control tasks.

The first task the agents are evaluated on is called Mountain Car. A sample image of the environment is presented on figure 2. The goal is to get the cart up the hill where the flag is. In order to get up the steep hill the agent has to first build momentum on the left hill.

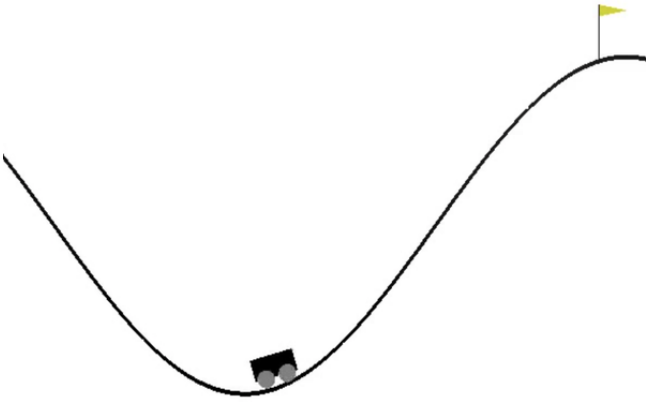

Fig. 2. Sample image from the Mountain Car task.

E. Task 2: Lunar Lander

Second task we use for evaluation is called Lunar Lander. A sample image of the environment is presented on figure 3. The goal is to land the vehicle between the 2 flags. Agent has control over 2 continuous variables which control the 2 axis of acceleration.

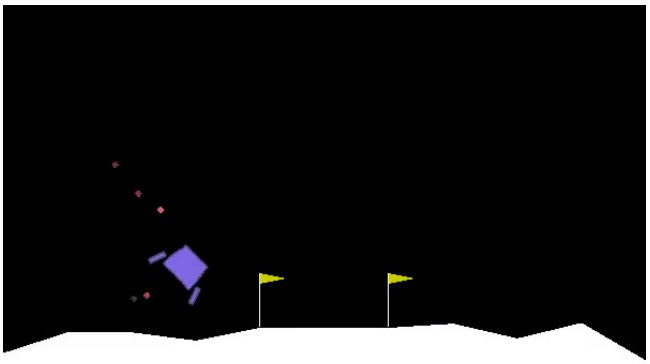

Fig. 3. Sample image from the Lunar Lander task.

F. Task 3: Bipedal Walker

Third task we use for evaluation is called Bipedal Walker. A sample image of the environment is presented on figure 4. The agent has to walk to the end of the polygon on the right side of the screen. Agent has control over the motor torque applied to the joints of the walker.

We compare the agents based on the final score the algorithm achieves. We also evaluate robustness of the agents looking at the standard deviation at each time-step. Agents are also compared based on the convergence time. Convergence time meaning the time it takes the agent to achieve the final policy.

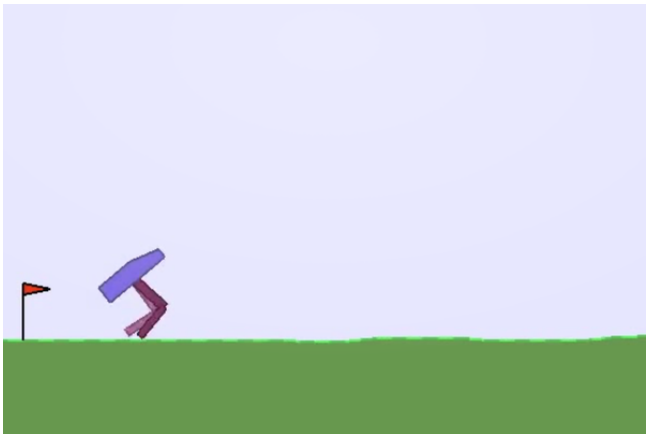

Fig. 4. Sample image from the Bipedal Walker task.

IV. Results

We evaluate the three reinforcement learning algorithms on three different continuous environments of ranging difficulties. For evaluation we used the OpenAI gym environment [3] and stable-baselines [10] for the algorithm implementations. In the table I we present results of the experiments ran on different tasks. We can see that SAC outperforms both A2C as well as PPO on two tasks based on the final reward it achieves and it does so by a margin of more than 200 points on the Bipedal Walker task. We also see that all agents come close to solving the Mountain Car task with PPO actually achieving the highest possible score of 0.

TABLE I
RESULTS OF THE EXPERIMENTS USING THREE DIFFERENT REINFORCEMENT LEARNING ALGORITHMS ON THREE DIFFERENT TASKS. HIGHER REWARD IS BETTER.

Task/Agent	A2C	PPO	SAC
Mountain Car	-5	**0**	-1
Lunar Lander	-218	113	**286**
Bipedal Walker	-79	-5	**206**

We plot a graph for each task performed by all 3 agents. On the graphs we have plotted the mean reward on each time-step (denoted by full line), and standard deviation over 5 runs (denoted by colored area around the mean line).

A. Mountain Car

From graph in figure 5 we observe that PPO achieves optimal final policy (with reward 0) on the Mountain Car task. Soft Actor Critic is the fastest to achieve the final policy with only around 10000 time-steps. It also achieves nearly optimal performance on the given task. Asynchronous Actor Critic gets stuck in local optimum and does not seem to find a better policy even after 10 thousand time-steps.

B. Lunar Lander

The performance of the agents on the Lunar Lander task is present on graph in figure 6. We observe that A2C and PPO fail to solve the task after one million time-steps, while SAC does so after about 300 thousand time-steps. Both A2C and PPO are more unstable (higher standard deviation) than SAC, making them less reliable of this particular task.

C. Bipedal Walker

The performance of the agents on the Bipedal Walker task is present on graph in figure 7. We observe that PPO and A2C fail to get the walker moving at all. Soft Actor Critic manages to get the walker moving and making some distance but it never gets 300 points as reward, meaning that the walker falls before reaching the finish line. We also observe that SAC is very unstable on this task compared to Lunar Lander and Mountain Car.

From the experiments we can infer that the Bipedal Walker is the hardest of the 3 tasks to solve for this type of RL algorithms. We also observe that there are clear differences in algorithm performance. Soft Actor Critic outperforms other two algorithms on two more complex tasks (Lunar Lander and Bipedal Walker) while reaching near optimal final policy on a simpler task (Mountain Car). Since the main difference between SAC and PPO is in the entropy regularization we can assume that efficient exploration of SAC is the main advantage of this algorithm.

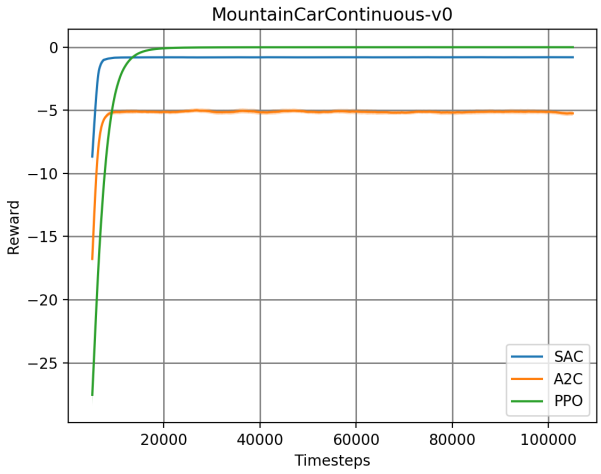

Fig. 5. Comparison of A2C, PPO and SAC on Mountain Car task.

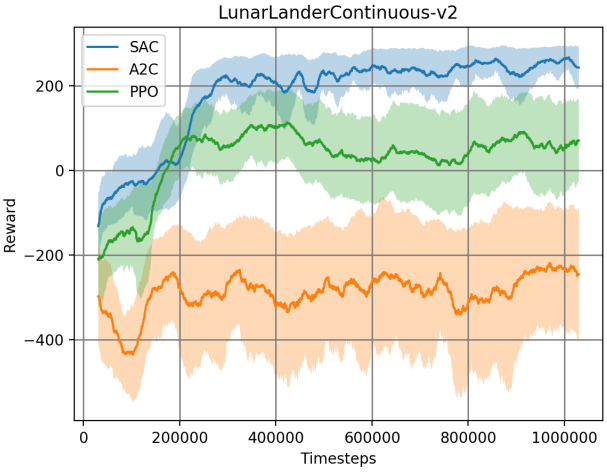

Fig. 6. Comparison of A2C, PPO and SAC on Lunar Lander task. We plot the reward agent receives on each timestamp.

V. Conclusion

Reinforcement learning has made a lot of progress in the recent years. Some of that progress is due to improved algorithms and we are interested how much those algorithms differ in performance. This kind of algorithms show great promise for solving real world tasks such as intractable optimization problems and continuous control. To evaluate reinforcement learning algorithms many simulated environment that serve as benchmarks have been created.

We make a brief review of related work where we list some promising ways of improving RL agents and maybe

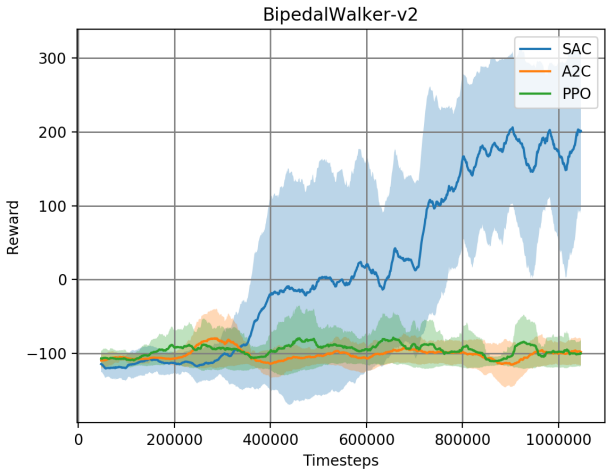

Fig. 7. Comparison of A2C, PPO and SAC on Bipedal Walker task.

even essential components of intelligent systems that can solve complex real world problems and can navigate in physical environment. In this paper we tested 3 model-free reinforcement learning algorithms and compared how well they perform on 3 different continuous control tasks. We evaluated their performance by looking at the time it takes each agent to reach the final policy and how high of the score the agent is able to achieve. We also looked at the reliability of each agent or how stable the agent's learning is given different initialization parameters.

We find out that Soft Actor Critic is outperforming both Proximal Policy Optimization and Asynchronous Actor Critic in terms of final reward achieved, robustness and time to converge. Proximal Policy Optimization has shown to be able to achieve higher results on simpler tasks like Mountain Car than A2C and Lunar Lander but it lacks good exploration strategy to solve Bipedal Walker and Lunar Lander completely. Even though PPO fails to learn how to solve presented tasks, it has been shown that PPO scales to the some of the most complex simulated environments such as Dota2 and is able to surpass human performance [1]. The downside of such methods is extremely large sample complexity that SAC is trying to solve by incorporating effective exploration. We also observed that there are large differences between the difficulty of the tasks with Bipedal Walker being the most difficult and Mountain Car being the simplest for the agents to solve.

In further work we could compare model-free RL algorithms and model-based RL algorithms. Comparing model-based where the model is given and the ones where the model has to be learned could also give us some sense of how much progress has been made on the sample efficiency and ability so solve complex tasks where planning ahead is essential. We could also test the RL algorithms on more different tasks containing some real world problems.

References

[1] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[2] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[5] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[6] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[8] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2577–2587, 2017.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[10] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[11] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[12] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[13] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[14] 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, 2016.

[15] 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.

[16] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[17] Tom Le Paine, Sergio Gómez Colmenarejo, Ziyu Wang, Scott Reed, Yusuf Aytar, Tobias Pfaff, Matt W Hoffman, Gabriel Barth-Maron, Serkan Cabi, David Budden, et al. One-shot high-fidelity imitation: Training large-scale deep nets with rl. *arXiv preprint arXiv:1810.05017*, 2018.

[18] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[19] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

[20] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2827–2836. JMLR. org, 2017.

[21] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

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

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

[24] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[25] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. Number 4. MIT press Cambridge, 1998.

[26] 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*, pages 1057–1063, 2000.

[27] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[28] Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Advances in neural information processing systems*, pages 3486–3494, 2016.

[29] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[30] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.