Conditioning of Reinforcement Learning Agents and its Policy Regularization Application

Arip Asadulaev [1] **Igor Kuznotsov** [1] **Gideon Stein** [1] **Andrey Filchenkov** [1]

Abstract

The role of conditioning was studied for supervised learning problems (Pennington et al., 2017). It also was shown that conditioning regularization can help to avoid the "mode-collapse" problem in Generative Adversarial Networks (Odena et al., 2018). In this paper, we try to answer the following question: **Can information about policy conditioning help to shape a more stable and general policy of reinforcement learning agents?** To answer this question, we conduct a study of conditioning behavior during policy optimization. To the best of our knowledge, this is the first work that research condition number in reinforcement learning agents. We propose a conditioning regularization algorithm and test its performance on the range of continuous control tasks. Finally, we compare algorithms on the CoinRun (Cobbe et al., 2019) environment with separated train end test levels to analyze how conditioning regularization contributes to agents' generalization.

1. Introduction

Generalization in Reinforcement Learning (RL) is different from supervised learning generalization problem (Zhang et al., 2018). We need specific techniques to avoid overfitting of RL algorithms (Farebrother et al., 2018). Agents can achieve different scores on the test set while all of them achieved the same rewards during training. In RL, the test data performance depends on agent architecture, because different architectures have different *priori* algorithmic preferences (inductive biases) (Zhang et al., 2018).

For example, Convolutional Neural Networks (CNNs) agents are too sensitive to small visual changes and can completely fail due to perturbations (Lee et al., 2020). Such techniques as the first CNNs layer randomization can avoid it and help to learn robust representations (Lee et al., 2020). **In our paper, to control agents sensitivity to small changes in the environment, we propose to use agent condition number regularization.**

Conditioning or **condition number** is the measure that indicates the proximity of a neural network to the Dynamical Isometry property. **Dynamical Isometry** is a neural network property stating that the distance between a network's inputs is the same as the distance between outputs. This property can be achieved by having a mean squared singular value equal to $\mathcal{O}(1)$ of a Jacobian input-output network (Pennington et al., 2017).

For classification problems, was shown that well-conditioned neural networks could significantly speed up training (Pennington et al., 2017). The role of conditioning was also studied for Generative Adversarial Networks (GANs) and it was shown that conditioning is causally related to the generator performance, and a conditioning regularization can help to avoid the "mode-collapse" problem (Odena et al., 2018).

Before using conditioning regularization in RL agents, we conduct a study of the relationship between policy performance and conditioning to find justifications for using it as a regularization. We analyze the behavior of the agent conditioning on different policies that are set by different sets of hyperparameters and see a correspondence between the conditioning and the ratio of achieved rewards.

Based on these observations, **we apply condition number regularization** to Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) and Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithms and compare their performance on 8 continuous control tasks in the PyBullet environment (Ellenberger, 2018). In our experiments, models with regularization outperformed other models on most of the tasks. To explicitly test how our regularization affects on agent generalization, we run the PPO algorithm with conditioning regularization on CoinRun environments(Cobbe et al., 2019). The results are presented in Section 3.

[1]ITMO University, Saint-Petersburg, Russia. Correspondence to: Arip Asadulaev <aripasadulaev@itmo.ru>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

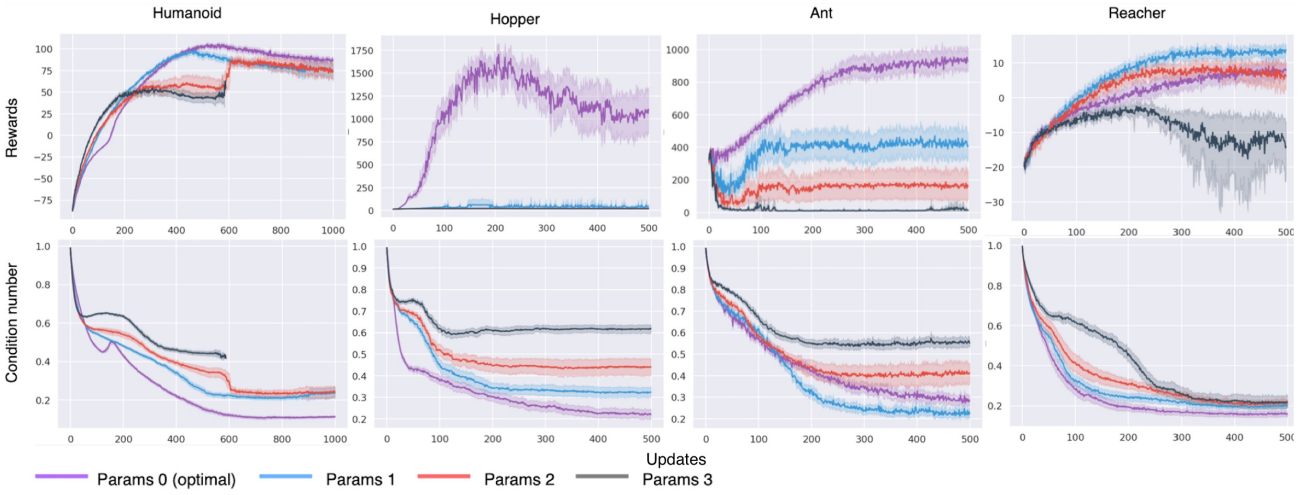

Figure 1. PPO rewards and conditioning ψ in PyBullet environments with different hyperparameters. Each curve is obtained by averaging results of 30 agents (10 for each seed)

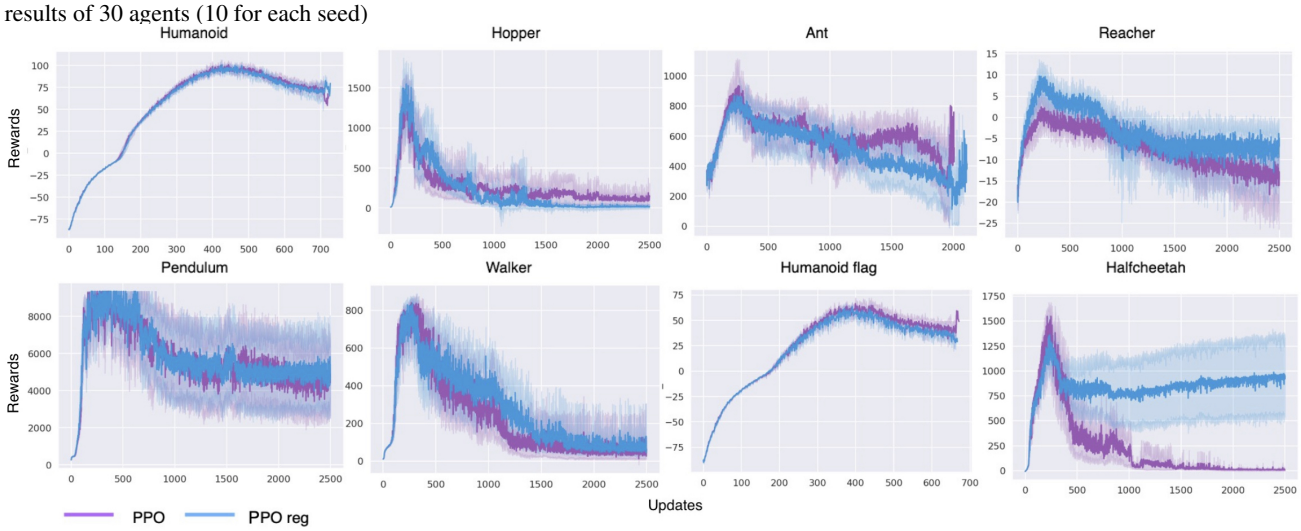

Figure 2. PPO rewards and conditioning ψ, in PyBullet environments. Each curve is obtained by averaging results of 30 agents (10 for each seed).

2. Conditioning of RL Agents

Conditioning Estimation: Getting the mean squared singular value of a Jacobian input-output network using Singular Value Decomposition (SVD) is time-consuming. Because of this, we adapted a technique designed to assess GAN models conditioning to RL agents for fast conditioning estimation. Jacobian Clamping (JC) (Odena et al., 2018) is an algorithm that computes the condition number of the generator's Jacobian and locks it inside the interval where the Dynamical Isometry property can be achieved.

To compute conditioning in RL agents, we feed two mini-batches at a time to the agent. The first batch consists of the real environment states S_t at timestep t. The second batch consists of the same states but with some added disturbance δ. Then we estimate how these batches affected the agent: $J_t = \frac{\|\pi_\theta(S_t) - \pi_\theta(S_t+\delta)\|}{\|\delta\|}$. After this, we compute the value ψ_t that characterizes how close J_t is to the range $\lambda_{\max}, \lambda_{\min}$. These values approximately set the desirable range for model conditioning. We saved these parameters equal to the range defined previously for GANs (1 and 20).

$$\begin{aligned} \psi_t^{max} &= \left(\max\left(J_t, \lambda_{\max}\right) - \lambda_{\max}\right)^2, \\ \psi_t^{min} &= \left(\min\left(J_t, \lambda_{\min}\right) - \lambda_{\min}\right)^2, \\ \psi_t &= \psi_t^{min} + \psi_t^{max}. \end{aligned} \tag{1}$$

More details are presented in Algorithm 1 in the Appendix.

Conditioning and Policy Performance: To examine the relationship between policy and conditioning, we run PPO with different hyperparameters and random seeds on the four continuous control tasks Humanoid-v0, Hopper-v0, Ant-v0, and Reacher-v0. Through these trials, we try to examine whether ineffective policies are less conditioned. We use the standard PPO parameters as the optimal configuration

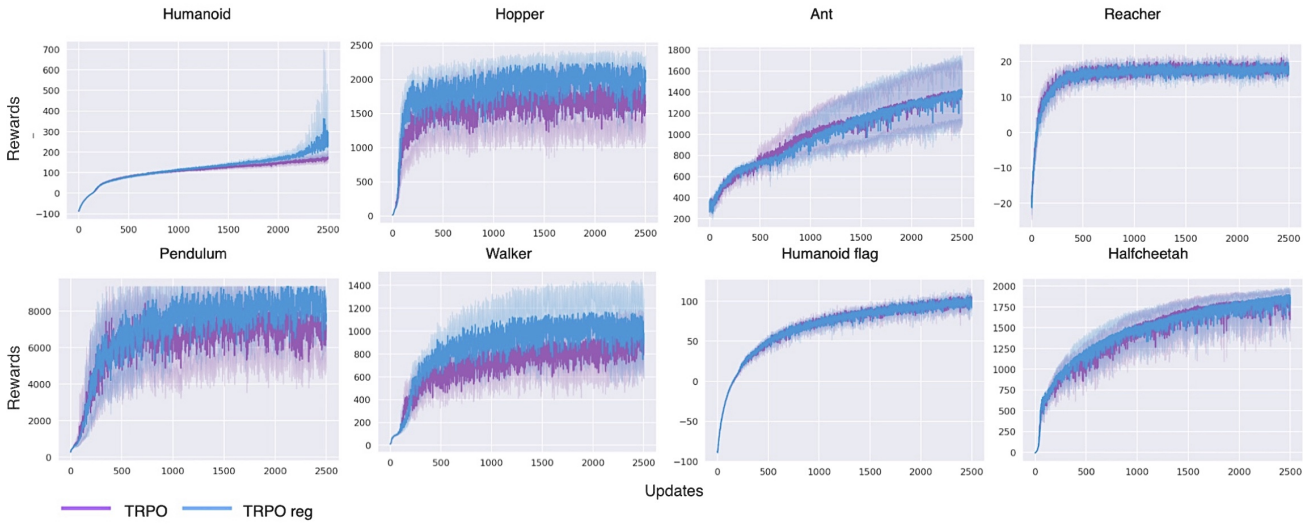

Figure 3. TRPO vs TRPO reg. Rewards and conditioning ψ, in PyBullet environments. Each curve is obtained by averaging results of 30 agents (10 for each seed)

Env	Humanoid	Hopper	Ant	Reacher	Pendulum	Walker	Humanoid Flag	HalfCheetah
PPO	73.1	106.8	414.1	-13.5	4534.9	55.1	43.6	9.9
PPO reg	73.2	18.6	294.8	-7.4	4978.4	74.2	38.4	941.2
TRPO	165.6	1697.0	1376.8	**18.2**	6993.6	892.4	**98.7**	1795.3
TRPO reg	**245.8**	**2001.2**	**1379.4**	17.7	**8213.8**	**1013.5**	**98.7**	**1843.3**

Table 1. Mean reward over the last 100 optimization steps for TRPO, PPO, PPO reg, and TRPO reg. The mean was computed over 3 random seeds and 10 agents for each seed using optimal policy hyperparameters

and made three adjustments to those parameters to produce less effective policies.

In each configuration, we use the same minibatch size, the number of timesteps T, PPO epoch, policy learning rate, and η. Parameters that were modified are: GAE parameter, discount (γ), value function (VF) coefficient, VF learning rate, VF epochs (Schulman et al., 2015; 2017). The sets of hyperparameters are presented in Table 3 in the Appendix. We test each setting on 4 PyBullet environments with 3 random seeds and 10 agents for each seed. Results are presented in Figure 1.

Results Discussion: Our experiments show that the conditioning has similar patterns with the number of received rewards. On the Humanoid task, we found that the most effective policy has the lowest conditioning. And furthermore, parameter 2 model drop of condition number corresponds to the moment of a sharp increase in rewards for the agent.

The connection between policy and conditioning is not clearly evident in the Ant task. However, agents with parameters 2 and 3 that obtained smaller reward values are more distant from the dynamical isometry property. Furthermore, an interesting observation that is worth noting is that policies, which are well-performing and gain higher reward values at the end of the training, are better conditioned, often even from the first training steps.

Because of environment dynamics, a linear relationship between the reward curves and the condition number is difficult to establish. However, in general, based on these experiments, a pattern can be observed: **a policy that receives fewer rewards has less optimal conditioning.** Also, turning back to the privileges that Dynamical Isometry provides for deep non-linear networks in classification and generation tasks too, we assume that if an agent is closer to Dynamical Isometry, it will allow forming a more stable and efficient policy.

3. Conditioning Regularized Policy Optimization

In this section, we propose an algorithm that regularizes the condition number of the agent. To regularize the policy we simply use the values of conditioning as a penalty. The example of regularized PPO presented below. We used the PPO algorithm and added a value of ψ to the surrogate policy loss:

$$L_t^{CLIP+\psi+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) + c_1\psi - c_2 L_t^{VF}(\theta) + c_3 S\left[\pi_\theta\right](s_t) \right], \quad (2)$$

where L^{CLIP} is PPO policy loss. c_1 is coefficient for conditioning penalty, L_t^{VF} is a value loss $\left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$

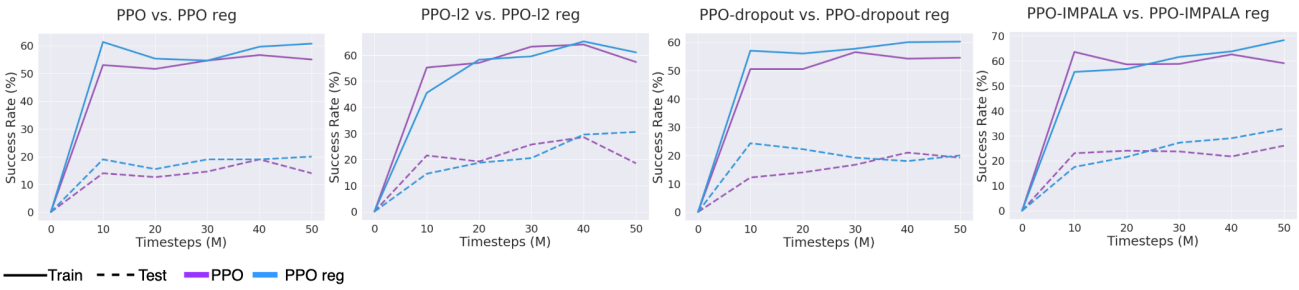

Figure 4. PPO and PPO with conditioning regularization. Success rate on CoinRun environment on train and test levels

Levels	PPO	PPO reg	PPO-l2	PPO-l2 reg	PPO-D	PPO-D reg	PPO-IMP	PPO-IMP reg
Seen	55	**60.7**	57.2	**61**	54.5	**60.2**	59	**68.2**
Unseen	14	**20**	18.5	**30.5**	19.2	**20**	26	**32.8**

Table 2. PPO and PPO with conditioning regularization. Success rate after 50M timesteps on CoinRun environment, on train(seen) and test(unseen) levels.

with coefficient c_2, $S\left[\pi_\theta\right]\left(s_t\right)$ is policy entropy for state s_t multiplied by entropy coefficient c_3. Conditioning penalty can be applied to other algorithms too, in our experiments we used it for TRPO as well. Condition value used for a penalty computing on the new policy on PPO and TRPO algorithm.

Continuous Control Experiments: We conduct experiments of the regularization technique on PPO and TRPO algorithms. We optimized 30 agents for each task (10 agents for 1 random seed) over 2500 updates (5 million timesteps) see Figure 2, 3. We test algorithms on Humanoid-v0, Hopper-v0, Ant-v0, Reacher-v0, Double Inverted-Pendulum-v0, Humanoid-Flag-v0, Walker-v0, and Half-cheetah-v0 environments.

In this test, the hyperparameters setting is equal to the optimal one, presented in PPO and TRPO literature (Schulman et al., 2015; 2017) for continuous control tasks. For the TRPO algorithm, we also used mean conditioning of a trajectory as a penalty for surrogate policy loss. In our experiments both basic TRPO and regularized one show better results than PPO. The average rewards for the last 100 updates are shown in Table 1. In all experiments model with name "reg" is conditioning regularized model. For experiments, we used the penalty multiplied by a coefficient c_1 equal to 0.001.

Generalization Experiments: Our continual learning problem was set without explicitly separated training and testing stages. In generalization experiments, we trained models on the fixed large-scale set of 500 levels of CoinRun (Cobbe et al., 2019) and tested on unseen levels. In this experiment, we run PPO with $l2$ and Dropout (Srivastava et al., 2014) regularizations, then we run the same methods but with additional conditioning penalty. For this experiment, we use "NatureCNNs" architecture proposed for tests in (Cobbe et al., 2019). Also, we tested the PPO method without $l2$ and dropout regularization but based on IMPALA (IMP) (Espeholt et al., 2018) architecture.

We noticed a high variance in scores during tests. Due to that, at evaluation, we increase the number of repeats form 5 as it was used in (Lee et al., 2020) to 20. We trained models over 50M timesteps, but only on one random seed, all other settings were equal to (Lee et al., 2020) (Section 4.2). Results are presented in Figure 4 and Table 2. Our method outperforms PPO in all 4 training scenarios.

4. Discussion and Future Work

In this work, we propose a simple and computationally inexpensive optimization method for Deep RL. We adapted a technique called Jacobian Clamping to approximately estimate conditioning of the agent. We tested our approach on the PyBullet and CoinRun domains. In our opinion, extending RL algorithms by conditioning regularization is a promising research direction. Condition number can provide important information about the policy, such as the correctness of hyperparameters or stability.

However, our work is still in progress. To study the role of conditioning for the generalization problem more thoroughly, we plan to conduct a test on the CoinRun environment with more timesteps and random seeds. Our experiments show that different architectures conditioning regularization produces various results. We plan to test conditioning contribution to other architectures too and run them on the environments like DeepMind Lab (Beattie et al., 2016). Also, we plan to compare conditioning regularization with other methods such as information bottleneck (Goyal et al., 2019; Galashov et al., 2019; Igl et al., 2019), . Estimating conditioning directly using the Jacobian matrix and SVD would be a very important experiment to examine conditioning in RL agents too.

References

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. Deepmind lab. *CoRR*, abs/1612.03801, 2016. URL http://arxiv.org/abs/1612.03801.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289. PMLR, 2019. URL http://proceedings.mlr.press/v97/cobbe19a.html.

Ellenberger, B. Open-source implementations of openai gym mujoco environments for use with the openai gym reinforcement learning research platform. https://github.com/benelot/pybullet-gym, 2018.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in DQN. *CoRR*, abs/1810.00123, 2018. URL http://arxiv.org/abs/1810.00123.

Galashov, A., Jayakumar, S. M., Hasenclever, L., Tirumala, D., Schwarz, J., Desjardins, G., Czarnecki, W. M., Teh, Y. W., Pascanu, R., and Heess, N. Information asymmetry in kl-regularized RL. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1lqMn05Ym.

Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Larochelle, H., Botvinick, M., Bengio, Y., and Levine, S. Infobot: Transfer and exploration via the information bottleneck. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJg8yhAqKm.

Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., and Hofmann, K. Generalization in reinforcement learning with selective noise injection and information bottleneck. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 13956–13968, 2019. URL https://arxiv.org/abs/1910.12911.

Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Are deep policy gradient algorithms truly policy gradient algorithms? *CoRR*, abs/1811.02553, 2018. URL http://arxiv.org/abs/1811.02553.

Lee, K., Lee, K., Shin, J., and Lee, H. Network randomization: A simple technique for generalization in deep reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJgcvJBFvB.

Odena, A., Buckman, J., Olsson, C., Brown, T., Olah, C., Raffel, C., and Goodfellow, I. Is generator conditioning causally related to GAN performance? 80:3849–3858, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/odena18a.html.

Pennington, J., Schoenholz, S. S., and Ganguli, S. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *CoRR*, abs/1711.04735, 2017. URL http://arxiv.org/abs/1711.04735.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014. URL http://dl.acm.org/citation.cfm?id=2670313.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *CoRR*,

abs/1804.06893, 2018. URL http://arxiv.org/abs/1804.06893.