PARAMETER SPACE NOISE FOR EXPLORATION: ICLR 2018 REPRODUCIBILITY CHALLENGE

Clayton Connors, Steven Schmatz & Bhairav Mehta *
Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48014, USA
{clconnor,sschmatz,bhairavm}@umich.edu

ABSTRACT

Deep reinforcement learning (RL) methods generally engage in exploratory behavior through noise injection in the action space. We analyze the validity of using parameter space noise as described in the paper *Parameter Space Noise for Exploration* (Anonymous (2018)). Using the baseline code provided at `https://github.com/openai/baselines`, we examine the method of noise injection described in the paper and aim to reproduce the results. We have detailed the reproducibility of the paper by testing its effectiveness on discrete and continuous action-spaced environments, using both on- and off-policy algorithms. Our results show partial reproducibility on the subset of experiments we have conducted.

1 INTRODUCTION

Exploration, with regards to deep reinforcement learning, is important to the performance of an agent as it helps ensure behavior does not converge prematurely to local optima. Most previous approaches, such as bootstrapped DQN (Mnih et al. (2013)), inject noise into the action space, leading to unpredictable exploration that is not correlated to the agent's parameters.

The paper under examination (Anonymous (2018)), which injects temporally-correlated noise directly into the parameter space, cites benefits of being not only able to scale to large function approximators such as neural networks, but also to employ the method to off-policy algorithms such as DQN, DDPG (Lillicrap et al. (2015)), and TRPO (Schulman et al. (2015)). The method extends upon previous work by forcing agents to explore their environment more effectively by making an agent's exploration more consistent across timesteps. By adding controlled noise to the parameter space, the paper shows empirical results on discrete and continuous control tasks, using both on- and off-policy algorithms, that surpass results shown by methods employing injection of action space noise.

The purpose of this study is to analyze the reproducibility of this paper. We ran experiments using parameter noise injection across a subset of the discrete and continuous environments. In addition, we ran ablation experiments, studying the impact of changing the parameter noise threshold on learning. Finally, we summarize the reproducibility of the paper in question.

2 RELATED WORK

The most related previous work was Rückstieß et al. (2008), which showed that parameter space noise improves exploration for policy gradient methods in low-dimensional state spaces. The parameter space noise paper extends this work to high-dimensional state spaces and policies in the on- and off-policy setting.

Shortly after the parameter space noise paper was released, there was an additional paper, Fortunato et al. (2017), which built upon the work in the parameter space noise paper by making the parameters

*All authors contributed equally.

of the noise an additional learned parameter of the network. It was found that doing so yielded substantially higher scores on a wide range of Atari games.

3 METHODOLOGY

In the following section, we describe the methods that will be used to attempt to recreate the experiments detailed from the original paper. From our analysis, we will be able to evaluate the reproducibility of the original work. We test the methods on both discrete and continuous control environments, using the Arcade Learning Environment (Bellemare et al. (2012)) and Roboschool Continuous Control Environments (`https://github.com/openai/roboschool`). We detail further hyperparameter and network architecture details below.

3.1 ARCADE LEARNING ENVIRONMENT

For the Arcade Learning Environment experiments, we used the network architecture described in Bellemare et al. (2012), consisting of convolutional layers followed by fully connected layers. In addition, some of our hyperparameters differed from the original study, following instead the settings of the OpenAI implementation. The architecture used along with our hyperparameter modifications are fully described in the appendix.

3.2 CONTINUOUS CONTROL ENVIRONMENTS

We also compare parameter noise with action noise on the continuous control environments implemented in OpenAI Gym (`https://github.com/openai/gym`). We use DDPG (Schulman et al. (2015)) as the RL algorithm for all environments with similar hyperparameters as outlined in the original paper except for the fact that layer normalization (Lei Ba et al. (2016)) is applied after each layer before the nonlinearity, which we found was shown in Anonymous (2018) to be useful in either case and especially important when utilizing parameter space noise.

Exactly like in the original paper, each agent is trained for 1M timesteps, where 1 epoch consists of ten thousand timesteps. In order to make results comparable between configurations, we evaluate the performance of the agent every ten thousand steps by using no noise for 20 episodes.

4 RESULTS

4.1 ARCADE LEARNING ENVIRONMENT

The OpenAI baseline implementation uses both prioritized experience replay and dueling networks, so they are used in our tests as well. Our results on Enduro, displayed in Figure 1, show parameter noise performing worse than epsilon-greedy exploration with these improvements and the hyperparameter configuration set in the implementation. Note that, in the results reported by OpenAI for their implementation (`https://github.com/openai/baselines-results/blob/2c78aefba7a7479cf7728a65c313879fa0f085da/param-noise/atari.md`), for most environments, epsilon-greedy exploration performs nearly identically to parameter noise when augmented with these two improvements. For Enduro, the point to note is that dueling networks and prioritized experience replay bring epsilon-greedy exploration up to the level reported for parameter noise in Anonymous (2018), but do not increase or decrease the performance of parameter noise beyond what was reported in that paper. This holds true both for our results and for the OpenAI results.

4.2 CONTINUOUS CONTROL ENVIRONMENTS

4.2.1 RESULTS USING DEEP DETERMINISTIC POLICY GRADIENTS

We also tested parameter noise on an open-source implementation of the deep deterministic policy gradient algorithm. The test compared parameter noise against both correlated action space noise generated by an Ornstein-Uhlenbeck process as described in Uhlenbeck & Ornstein (1930) and uncorrelated action noise. As a baseline, we also tested a configuration with no noise injection

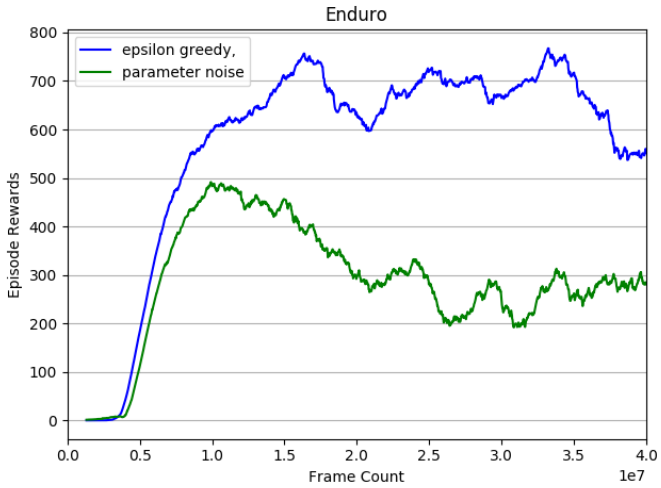

Figure 1: Epsilon-greedy exploration versus parameter noise on Enduro.

method at all. Owing to time constraints, we were unable to complete the full run of this experiment. As a result, we cannot present a qualitative analysis of parameter noise for continuous environments.

Qualitatively, we confirmed similar visual results as the original paper within the span of the experiment which we were able to complete. On the continuous control environment `HalfCheetah`, action space noise methods converge to a local optimum, and the agent learns to flip to its back and wiggle its way forward. With parameter space noise, the agent learns to break out of this local optimum, and through continued exploration, it learns a more realistic, higher-rewarding gait.

4.3 Ablation studies

In our ablation studies we found that using layer normalization and choosing the right standard deviation threshold drastically improve performance. For more details, see the appendix.

5 Reproducibility Conclusions

From the set of experiments that we have reproduced, we have observed that parameter space noise can cause qualitative improvements to policies in continuous action-space domains. However, our results on the discrete Enduro environment, together with the external results cited in section 4.1 on Enduro and other ALE environments, imply that parameter space noise does not necessarily confer any additional benefit to a network which already makes use of the dueling and prioritized replay improvements. The experimental setup was simple, owing to the existence of an open-source implementation with good documentation. We were only able to reproduce a small subset of the experiments due to the large amount of compute needed.

References

Anonymous. Parameter space noise for exploration. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ByBAl2eAZ`.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012. URL `http://arxiv.org/abs/1207.4708`.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. *CoRR*, abs/1706.10295, 2017. URL `http://arxiv.org/abs/1706.10295`.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *ArXiv e-prints*, July 2016.

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. *CoRR*, abs/1509.02971, 2015. URL http://arxiv.org/abs/1509.02971.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. *State-Dependent Exploration for Policy Gradient Methods*, pp. 234–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-87481-2. doi: 10.1007/978-3-540-87481-2_16. URL https://doi.org/10.1007/978-3-540-87481-2_16.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930. doi: 10.1103/PhysRev.36.823. URL https://link.aps.org/doi/10.1103/PhysRev.36.823.

6 APPENDIX

6.1 NETWORK ARCHITECTURE FOR ATARI LEARNING ENVIRONMENT

The convolutional layers, in order, have 32 filters of size 8 x 8 and stride 4, 64 filters of size 4 x 4 and stride 2, and 64 filters of size 3 x 3 and stride 1. The fully connected layers, in order, have 256 units (as defined in the OpenAI implementation) and a single output unit per action for that action's Q estimate. All layers except for the output layer use ReLU activation. The fully connected portion of the network uses layer normalization as described in Lei Ba et al. (2016). We train using the Adam optimizer (Lei Ba et al. (2016)) with a learning rate of 1e-4. The concatenation of the last four frames, each downsampled to 84x84 pixels and a single average color channel, is passed into the network as input, and each action chosen by the network is repeated four times. We use up to 30 noop actions at the beginning of each episode. The discount rate is set to 0.99 and rewards are clipped to the range [-1, 1].

6.2 HYPERPARAMETERS FOR ATARI LEARNING ENVIRONMENT

The following hyperparameters differ from those of the paper, as they instead follow the settings of the OpenAI implementation. We use a batch size of 32 over a replay buffer of size 10K. For the epsilon greedy policy, we linearly anneal epsilon from 1 to 0.01 over the first 4M timesteps. We run a single gradient descent update every 16 frames and perform 40K random actions before training starts. There is no separate policy head, as it had previously been found to be unnecessary when using adaptively scaled noise. Both the additional 0.01 epsilon-greedy action selection when using parameter noise and the gradient clipping at the output layer of Q were omitted because the implementation does not include these, potentially as they are not necessary when using adaptively-scaled noise. We also use prioritized experience replay and dueling networks, as these are enabled by default in the OpenAI implementation and reflected in their own reported results (mentioned in section 4.1).

6.3 RESULTS OF ABLATION STUDIES

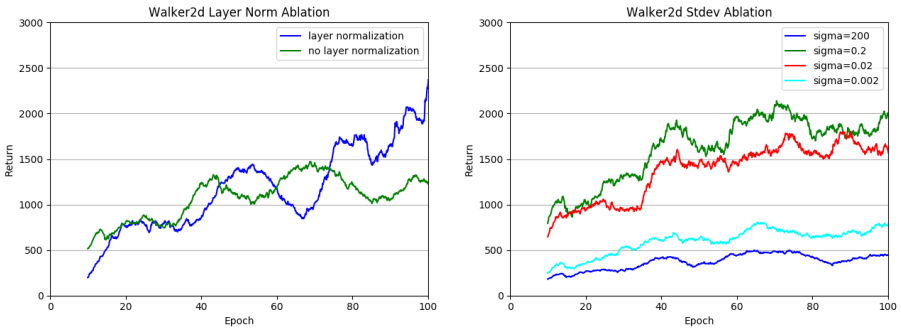

Figure 2: Ablation studies varying layer normalization and scale parameter.

To further inspect the reproducibility of this method, we turn to two key points presented in the original submission: *layer normalization* and the *scale parameter* σ. To study the effects of each, we ran configurations that varied both of these parameters on the MuJoCo environment *Walker2d*.

The results of the layer normalization test give some evidence for the necessity of layer normalization, although a comparison averaged over multiple runs would show this point or its negation more strongly. In addition, we show that the scale parameter σ is a carefully tuned hyperparameter, and when it is too large or small, policy improvement stops almost entirely.