International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 10 (2018) pp. 7977-7982

Dropout Acts as Auxiliary Exploration

Tegg Taekyong Sung[1], Daeyeol Kim[1], Soo Jun Park[2], and Chae-Bong Sohn[1]

[1]Department of Electronics and Communications Engineering, Kwangwoon University
20, Gwangun-ro, Nowon-gu, Seoul, 01897, Republic of Korea.

[2]Bio-Medical IT Convergence Research Department, ETRI
218, Gajeong-ro, Yuseong-gu, Daejeon, 34129, Republic of Korea.

Abstract

Deep neural networks have successfully been used in machine learning field, and scientists have been experimented that one of its methods, reinforcement learning is corresponded to the functions of basal ganglia in the brain. One of the critical issues in reinforcement learning is performing the optimal action for an agent. Commonly, this can be achieved by balancing between exploitation and exploration. Recently, dropout, one of the stochastic regularization methods, can be worked for discovering exploration. In this paper, we extend dropout as an auxiliary exploration in reinforcement learning, especially in continuous action problems. This method can be easily applied to any algorithms involving function approximator. We have empirically found the optimal dropout rates and position from layers in neural networks. Comparing to standard networks, dropout applied layers achieved higher rewards in most control tasks. Moreover, we suggest a promising methodology for developing dropout method using the probabilistic switch. With its probabilistic behavior, this can be attached to neuromorphic chip to perform dropout.

Keywords: Deep Learning, Reinforcement Learning, Improving Exploration, Continuous Control, Probabilistic Switch

INTRODUCTION

A newborn baby cannot grab objects accurately, but when he grows up, gathering plenty of experiences, reaches and grabs objects are an easy task. Such experiences starting from random trials to considerate action link to the chain of actions and behave as an expert. At each process, they produce consecutive rewards and the goal of every task is to achieve the maximum rewards as possible. In the computer science field, this activity is related to reinforcement learning which is learning interactions with the environment [1]. Recently, reinforcement learning uses deep neural networks (DNNs) as a function approximator and increases agent performances dramatically. As results, the deep reinforcement learning has solved many tasks including games [2], information extraction [3], visual tracking [4], robotics [5], and more.

Referred to human behavior, the trade-off between the need to obtain new knowledge and the need to use that knowledge to improve performance is one of the most critical issues to perform optimal behavior [6]. An agent can select either a maximized rewarding action at present state (exploitation) or a random action for seeking better rewarding action in future instead (exploration). If an agent takes explorations more than exploitations or vice versa, then one can miss optimal behavior. Therefore, to achieve high rewards, this dilemma should be properly balanced.

Previously, a heuristic method such as epsilon-greedy search has been widely used to improve exploration [7]; more recently, perturbing parameters in action spaces to maintain exploration has been proposed [8,9]. Inspired that, we also perturb network structures by applying stochastic regularization which performs as an auxiliary exploration. In this paper, we have used dropout, constructing the random sub-networks from the original networks [10]. The nodes from networks are dropped out and act as a noise for input, which can be viewed as an uncertainty in deep learning. When agent performing environment with this information, one can properly decide when to exploit and to explore [11].

Extending Gal's work [11], we have experimented with auxiliary regularization to recent reinforcement algorithms, Deep Deterministic Policy Gradient (DDPG) [12] and Trust Region Policy Optimization (TRPO) [13] and the method has outperformed the standard neural networks, achieving higher rewards in continuous control tasks. In addition, we discovered the relationship between dropout and probabilistic switch [14] which is promising methodology by cooperating hardware chip.

RELATED WORKS

The epsilon-greedy search is a heuristic and state-independent exploration method which is mostly selecting a greedy action and a random action at small probability. Likewise, Thompson sampling or Boltzmann sampling is used to select a random action. Also, different approaches such as count-based [15], variational inference [16], and evolution strategies [17] have been researched for efficient exploration.

In addition, exploration can be achieved by dithering perturbation. Recent works have intentionally added noise distribution to parameter spaces to maintain random actions. Fortunato et al. [8] and Plappert et al. [9] have researched using discrete and continuous tasks respectively.

Previously, Gal et al [11] applied dropout at every layer and use Thompson sampling instead of epsilon-greedy to drive exploration. However, this work only applied to discrete tasks using Deep Q-Networks (DQN) and viewed as Bayesian probability perspective.

BACKGROUNDS

In reinforcement learning, an agent interacts with the environment and receives the reward. To maximize rewards, the agent continuously updates the model which is the performing actions. Scientists have been found that the model of reinforcement learning is closely related to the human brain activities. The temporal difference (TD) error which is the key ingredient to update better action is matched to the functions of the basal ganglia [18]: dopamine neurons [19,20], dopamine-dependent plasticity in the striatum [21,22], and action value represented by striatal neurons [23].

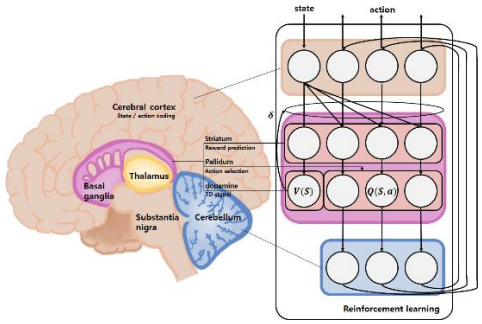

Figure 1: A schematic model of basal ganglia corresponded to the keywords of reinforcement learning.

Figure 1 is shown the correspondence of different areas of basal ganglia to the models of reinforcement learning, where $V(s)$ or $Q(s,a)$ represents reward when an agent has reached present state s or has performed an action a at state s, and δ represents the TD error. The signal of firing dopamine neurons in the brain corresponds to the TD error, and the TD error can be used to update the next performing action in reinforcement learning.

A. *Continuous control*

One of the widely known reinforcement algorithms, DQN has applied to a game, Go, and won the top human player [2]. However, naively applying DQN to continuous action tasks, such as waiving robot arms or driving autonomous driving car, cannot fully characterize agent movements. Degrees of their actions are much higher than that of discrete cases, causing the curse of dimensionality and very high cost of computations. Consequently, instead of using value function at each separate time step, directly optimizing the policy or trajectory of agent would be a more convincing approach. This method is called a policy optimization, and the two general examples are Deep Deterministic Policy Gradient (DDPG) and Trust Region Policy Optimization (TRPO).

B. *Deep deterministic policy gradient*

DDPG is an off-policy actor-critic algorithm that two policies are separately performed, discovering optimal policy and following existed policy [12]. It operates with two separate networks that actor follows policy and achieves a gradient of policy by sampling actions, while critic estimates Q-value functions with the gradient. The recursive Bellman equation used in actor-network is followed:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma Q\big(s_{t+1}, \pi_\theta(s_{t+1})\big), \qquad (1)$$

where π_θ is the policy. The parameters in this policy gradually reached to optimal by backpropagating through both actor and critic networks. For DDPG, *Ornstein-Uhlenbeck* noise is explicitly added to policy networks (actor) for attaining exploration. $\hat{\pi}_\theta(s_t) = \pi_\theta(s_t) + w$, where $w \sim OU(0, \sigma^2)$. This additional noise brings the network to select a random action at its related distribution.

C. *Trust region policy optimization*

In contrast to the off-policy method, TRPO is an on-policy algorithm that updating function approximator while following ongoing policy [13]. Simply, this is a combined method of traditional policy gradient [24] and natural gradient [25]. Estimating the gradient of expected return $\nabla_\theta \eta(\pi_\theta)$ via the likelihood ratio trick:

$$\nabla_\theta \eta(\pi_\theta) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i)(R_t^i - b_b^i), \qquad (2)$$

where N is the number of episodes, $R_t^i = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}^i$, and b_b^i is a baseline to reduce variance [26].

Hereafter, gradient descent algorithm performs under a certain boundary, trust region to avoid falling to local minima. The boundary is constructed using KL divergence. Second order derivatives is used and the algorithm takes a maximum step in this region. In each epoch, TRPO solves following evaluation:

$$maximize_\theta \quad E_{s\sim\rho_{\theta'}, a\sim\pi_{\theta'}}\left[\frac{\pi_\theta(a|s)}{\pi'_\theta(a|s)} A_{\theta'}(s,a)\right],$$
$$s.t. \quad E_{s\sim\rho_{\theta'}}[D_{KL}(\pi_{\theta'}(\cdot|s) \,\|\, \pi_\theta(\cdot|s))] \le \delta_{KL}, \qquad (3)$$

where ρ_θ is the discounted state-visitation frequencies, $A_{\theta'}(s,a)$ is the advantage value, and δ_{KL} is a step size parameter representing how much the policy is to change per iteration. In TRPO, instead of the empirical return, the advantage value that is the baseline subtracted from the return is used to reduce the variance of policy method.

D. *Dropout*

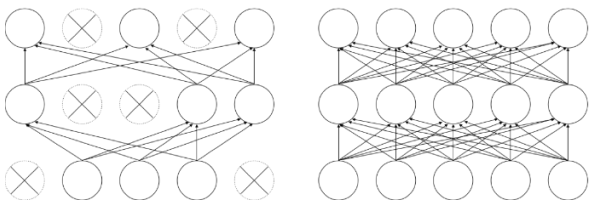

Figure 2: Dropout neural network model. **Left**: A standard neural networks. **Right**: A dropout applied neural networks. Crossed units have been dropped.

This method is one of the stochastic regularizations commonly used in the neural networks field. As shown in figure 2, nodes in standard neural networks are randomly disconnected at every training steps, generating different sub-networks. The drop rate is the probability depended on Bernoulli distribution, and information only passes through connected paths. Accordingly, the learning patterns would be different, and the networks are more tended to avoid existed patterns. When testing, all nodes are connected, and weights are fed. With this method, overfitting can be reduced, and the more generalized outcome could be achieved [10].

E. Probabilistic Switch

Probabilistic switch is a probabilistic system-on-a-chip (PSOC) architecture based on probabilistic algorithms. CMOS scaled into the nanometer regime has posed problems such as noise, parameter variations, or device perturbations. Probabilistic CMOS (PCMOS) is another design paradigm shifted to probabilistic design. The central idea of PSOC is to harness the probabilistic behavior of PCMOS devices to design architectural primitives with well-defined statistical behaviors [14,27]. In the probabilistic switch, instead of deterministic and uniform operations, probability parameter for the correctness p is involved. Table 1 has shown its probability functions.

Table 1: Probabilistic one-bit switching functions.

Input	Output	
0	0 (p)	1 $(1-p)$
1	1 (p)	0 $(1-p)$

(a) Identity function

Input	Output	
0	1 (p)	0 $(1-p)$
1	0 (p)	1 $(1-p)$

(b) Complement function

Input	Output	
0	0 (p)	1 $(1-p)$
1	0 (p)	1 $(1-p)$

(c) Constant function

Input	Output	
0	1 (p)	0 $(1-p)$
1	1 (p)	0 $(1-p)$

(d) Constant function

Figure 3 has shown the PCMOS switch. In practical, additional thermal noise is coupled to the output to generate noise and produce output as Gaussian distribution.

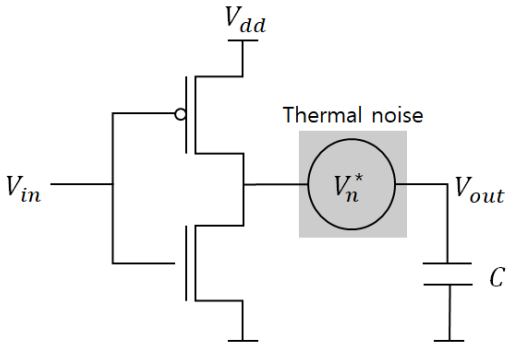

Figure 3: Probability switch.

The energy performance product (EPP) is a metric comparing architectural implementation related to both energy consumption and execution time that considers tradeoff performance for energy efficiency. The EPP gain, $\Gamma_{\mathcal{J}}$ is

$$\Gamma_{\mathcal{J}} = \frac{Energy_{\beta} \times Time_{\beta}}{Energy_{\mathcal{J}} \times Time_{\mathcal{J}}}, \quad (4)$$

where β is the EPP of the baseline and $\mathcal{J}$ is the EPP of the particular architectural implementation. Analyzing this gain, we get

$$\Gamma_{\mathcal{J}} = \left(1 + \frac{\mathcal{F} \times Energy_{flux,\beta}}{Energy_{cycle,host}}\right) \times \left(1 + \frac{\mathcal{F} \times Time_{flux,\beta}}{Time_{cycle,host}}\right), \quad (5)$$

PROPOSED METHODS

In the deep reinforcement learning, neural networks are used in policy as a function approximator. Considering how this regularization performs, the dropout applied networks act as perturbing networks that could sample random variable under Bernoulli distribution. Accordingly, an agent selects a random action under dropout applied networks. As the growth of using DNNs in reinforcement algorithms, this dropout method can be extensively applicable.

On the work of Gal et al. [11], a similar method has been researched. It uses dropout to include uncertainty in reinforcement learning. However, this only covers deterministic algorithms and approaches to probabilistic perspective. Extending that, we discovered dropout also can be applicable in continuous tasks and viewed the dropout usage as perturbing function approximation networks to discover additional exploration. Similar to dithering perturbation, by randomly constructing policy sub-networks using dropout sample random action values are executed. We empirically found the optimal dropout rate and its position in layers of neural networks. Dropout could directly be applied to any neural networks using policy optimization. In actor-critic networks such as DDPG, we only applied dropout to actor-network because exploration depends on action space noise.

Figure 4 has illustrated the design of general applications using probabilistic switches. One of its applications, the Bayesian inference is an application harnessing probabilistic behavior. Using equation 4, the EPP gain for Bayesian inference is 3. With this gain, the Bayesian inference can be developed using 7 probabilistic switches [14].

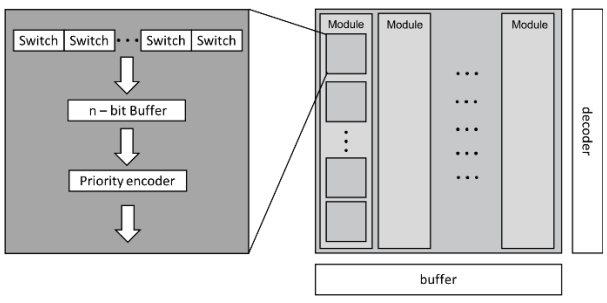

Figure 4: The applications design using PSOC.

Using equation 5, the variation of EPP gain with flux can be shown in figure 5.

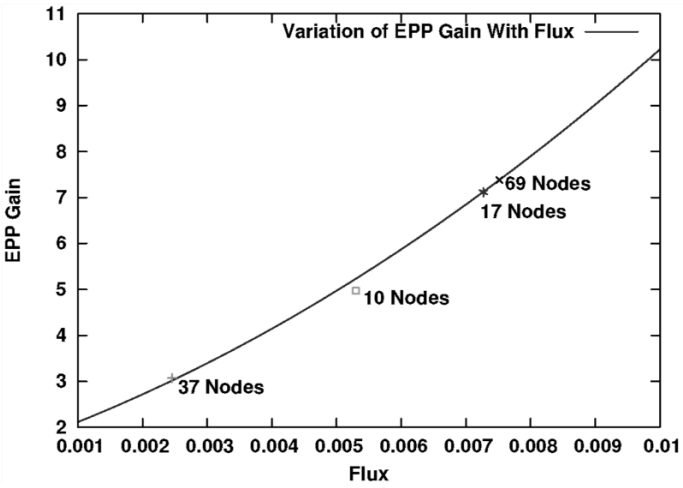

Figure 5: Variation of gain with respect to flux for a Bayesian network. Reprinted from Bilge et al's work [27].

In this application, probabilistic switches are implemented as coprocessors for acting probabilistic algorithm. We have seen that the probabilistic behavior of PCMOS can be attached to neuromorphic chips to replace dropout mechanism in neural networks. By this methodology, efficient regularization to networks can be implemented.

EXPERIMENTS

As the importance of the reproducibility of reinforcement algorithms, we use code by rllab [26]. As shown in figure 6, we simulate agents upon MuJoCo simulator [28]. The objective of each agent is to gait and to hop as fast as possible. Inputs are a position, angle, velocity, and angular velocity of thigh, leg, and foot.

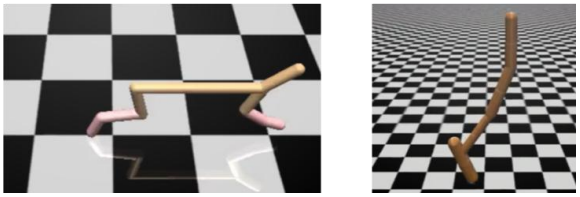

Figure 6: MuJoCo environments. **Left**: HalfCheetah. **Right**: Hopper.

We have experimented for the effects of dropout regularization in reinforcement algorithm. In the result graphs shown below, each x and y-axes are represented as average rewards and epochs respectively. We have performed 100 epochs and ran 5 models to gather ensembled results.

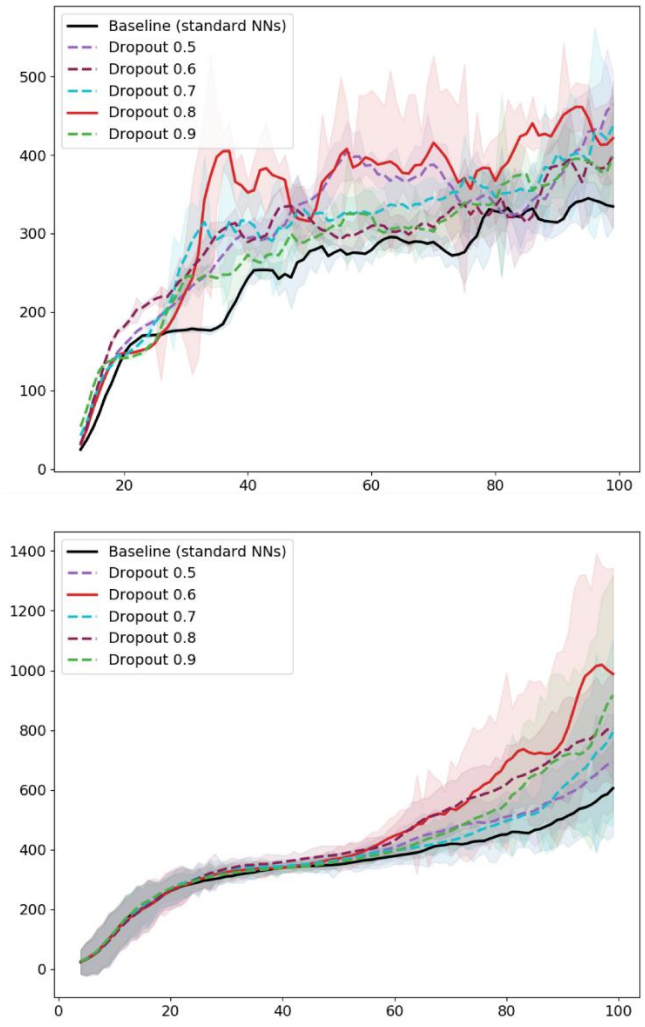

Figure 7: Results of applying different dropout rates using Hopper simulation. **Top**: DDPG algorithm and **Bottom**: TRPO algorithm are used.

In figure 7, we have used different dropout rates applied in standard neural networks to demonstrate dropout regularization method to be valid. The larger dropout rate, the lower probability of dropping nodes out in neural networks. Here, DDPG and TRPO algorithms are used respectively. The We performed using Hopper environment, and rates below than 0.5 are discarded, because they select more exploration than exploitation, resulting in inefficient. Dropout rates at 0.5 and at 0.8 have received the highest rewards in DDPG and TRPO respectively.

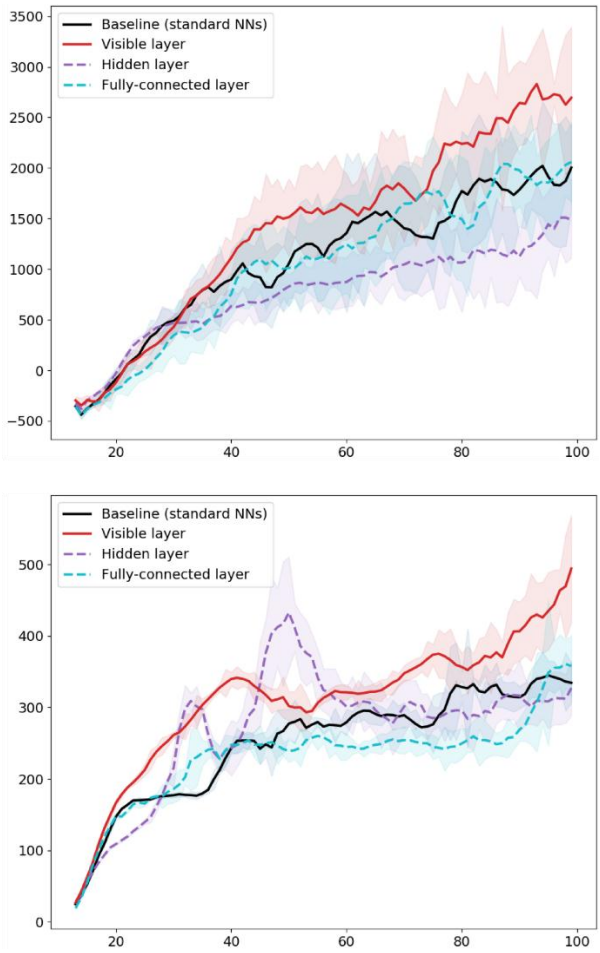

Figure 8: Results of applying dropout at different locations of a network layer. DDPG algorithm is used and experimented with different environments. **Top**: HalfCheetah and **Bottom**: Hopper are used.

In figure 8, different approaches of applying dropout in visible, hidden, and fully-connected layers are shown. From the previous results, we set dropout rate at 0.8 and simulated using two different environments. According to this analysis, applying dropout in the visible layer has achieved the highest rewards.

CONCLUSION

In this paper, we discovered an auxiliary exploration using dropout regularization. This can extensively applicable in any reinforcement algorithms that use neural networks as a function approximator. By perturbing standard networks to generate sub-networks, it can improve exploration by regularization. Utilizing network regularization, this method easily brings auxiliary exploration. We experimented DDPG and TRPO algorithms with MuJoCo environments, especially targeting to continuous control tasks. As results, dropout applied networks have outperformed than standard networks. Moreover, a close relationship between the dropout and the probabilistic switch as acting statistically on and off can offer promising scheme.

ACKNOWLEDGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2016-0-00288) supervised by the IITP (Institute for Information & communications Technology Promotion). The present research has been conducted by the research grant of Kwangwoon university in 2017.

REFERENCES

[1] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv*:1708.05866.

[2] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., and Dieleman, S. (2016) Mastering the game of Go with deep neural networks and tree search. *Nature*, 7587:484-489.

[3] Narasimhan, K., Yala, A., and Barzilay, R. (2016) Improving information extraction by acquiring external evidence with reinforcement learning. *arXiv preprint arXiv*:1603.07954.

[4] Choi, J., Kwon, J., and Lee, K. M. (2017) Visual tracking by reinforced decision making. *preprint arXiv*:1702.06291.

[5] Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016) End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 39:1- 40.

[6] Berger-Tal, O., Nathan, J., Meron, E., and Saltz, D. (2014) The exploration-exploitation dilemma: A multidisciplinary framework. *PloS one*, 9(4), e95693.

[7] Sutton, R. S., and Barto, A. G. (1998) Reinforcement learning: An introduction. *Cambridge: MIT press*.

[8] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., and Blundell, C. (2017) Noisy networks for exploration. *preprint arXiv*:1706.10295.

[9] Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. (2017) Parameter space noise for exploration. *preprint arXiv*:1706.01905.

[10] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014) Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 1:1929-1958.

[11] Gal, Y. (2016) Uncertainty in deep learning. *PhD thesis*, University of Cambridge.

[12] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015) Continuous control with deep reinforcement learning. *arXiv preprint arXiv*:1509.02971.

[13] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015) Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 15:1889-1897.

[14] Chakrapani, L. N., Pinar, K., Akgul, B. E., and Palem, K. V. (2007) Probabilistic system-on-a-chip architectures. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12.3-29.

[15] Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2017) #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2750-2759.

[16] Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016) Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, 1109-1117.

[17] Salimans, T., Ho, J., Chen, X., and Sutskever, I. (2017) Evolution strategies as a scalable alternative to reinforcement learning. *preprint arXiv*:1703.03864.

[18] Doya, K. (2007) Reinforcement learning: Computational theory and biological mechanisms. *HFSP Journal*, 1(1), 30.

[19] Schultz, W. (1995) Reward-related signals carried by dopamine neurons. In *Models of Information Processing in the Basal Ganglia*, Cambridge, Mass. 233-248.

[20] Schultz, W. (1998) Predictive reward signal of dopamine neurons. *Journal of neurophysiology*, 80(1) 1-27.

[21] Reynolds, J. N., Hyland, B. I., and Wickens, J. R. (2001) A cellular mechanism of reward-related learning. *Nature*, 413(6851), 67-70.

[22] Reynolds, J. N., and Wickens, J. R. (2002) Dopamine-dependent plasticity of corticostriatal synapses. *Neural networks*, 15(4-6), 507-521.

[23] Samejima, J., Ueda, Y., Doya, K., and Kimura, M. (2005) Representation of action-specific reward values in the striatum. *Science*, 310(5752), 1337- 1340.

[24] Williams, R. J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement learning*, 5-32.

[25] Peters, J., and Schaal, S. (2008) Natural actor-critic. *Neurocomputing*, 71(7-9), 1180-1190.
[26] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016) Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 1329-1338.

[27] Bilge, E. S. A., Lakshmi, N. C., Pinar, K., and Krishna, V. P. (2006) Probabilistic CMOS technology: a survey and future directions. In *Very Large Scale Integration, 2006, IFIP International Conference on IEEE*, 1-6.

[28] Todorov, E., Erez, T., and Tassa, Y. (2012) MuJoCo: A physics engine for model-based control. *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference*, 5026-5033.