Extensive Benchmark Evaluation of DroQ and Advanced Dropout Variant

Ann-Katrine Grodt Christiansen
ankagc@berkeley.edu

Jesper Hauch
jesperhauch@berkeley.edu

Abstract

Researchers developing reinforcement learning algorithms rely on evaluating their work on a handful of benchmark MuJoCo environments, which allow them to compare performance to other state-of-the-art algorithms. The MuJoCo environments are not sufficiently diverse to induce interesting learned behaviors from the real world (Henderson et al. [10]). The best option currently available is to evaluate the algorithms on a larger set of environments. The DroQ algorithm (Hiraoka et al. [11]) has shown impressive performance and high computational efficiency on four benchmark MuJoCo environments: Ant, Hopper, Humanoid, and Walker2d. DroQ is based on the randomized ensemble double Q-learning (REDQ) algorithm and uses a smaller Q-function ensemble with layer normalization and dropout to inject uncertainty into the model. We conduct an experiment on DroQ's performance by including four additional continuous-action space environments: HalfCheetah, LunarLanderContinuous, Pendulum, and Reacher. It was found that the environment features of state dimension, action dimension, episode length, and empirical task solvability scores, measured by Policy-Optimal Information Capacity and algorithm-based normalized task scores (Furuta et al. [7]), could not explain or find patterns in the performance of DroQ. A sparsity threshold λ was used to make the reward structure of environments sparse, increasing the difficulty for the agent trying to learn the environments (Amin et al. [1]). We showed that DroQ struggled to get reward in the sparse reward environments compared to the REDQ benchmarks. We propose DroQ$_{AD}$, a variant of DroQ, that replaces dropout in the Q-functions with advanced dropout (Xie et al. [20]). DroQ$_{AD}$ outperformed DroQ, REDQ, and SAC in the sparse reward environments but had a subpar performance for the majority of the eight other environments. Advanced dropout is a model-free dropout method that has shown superior results to other dropout methods in deep learning computer vision tasks such as MNIST and CIFAR-10 (Xie et al. [20]. The model-free methodology uses a reduction hyperparameter d, that controls the amount of preserved information from the input features used to model the dropout masks. The lack of computational resources is a major limitation of this paper, which entailed that significant results from having multiple runs of an algorithm could not be achieved.

1 Introduction

The performance of newly developed reinforcement learning (RL) algorithms is often evaluated on a handful of benchmark environments, to show comparable performance to the current state-of-the-art algorithms. The highly optimized MuJoCo environments (Todorov et al. [19]) are a major driver of the progress of benchmarks, due to their availability and adaptability to fit any possible model (Zhang et al. [21]), (Körber et al. [16]). Even though, MuJoCo simulates the basic physical dynamics of the world using multiple tunable parameters, Henderson et al. [10] find that existing environment simulators including MuJoCo are not sufficiently diverse, to induce interesting learned behaviors from the real world, or the algorithms are not sufficiently robust to cope with it. In the absence of environments that fulfill this, the best currently available option is to evaluate algorithms across a large set of environments, which are chosen in a way that imitates diversity.

An instance where this has not been conducted yet is with the recently published DroQ algorithm (Hiraoka et al. [11]). DroQ is proposed to make randomized ensemble double Q-learning (REDQ) (Chen et al. [5]) more computationally efficient while maintaining similar performance. The proposition replaced the large ensemble of Q-functions in REDQ with a smaller ensemble of Q-functions with dropout (Srivastava et al. [18]) and layer norm (Ba et al. [3]). Hiraoka et al. [11] find that DroQ has better or similar performance to REDQ in the benchmark MuJoCo environments, making it seem favorable for the maximum entropy reinforcement learning tasks considered.

We show that DroQ maintains high performance when evaluated on four additional environments. However, we find that performance for two of the benchmark environments does not correspond with the results of Hiraoka et al. [11]. Additionally, our analysis finds that the performance of DroQ breaks compared to REDQ when evaluated on an environments with a sparse reward structure. In this paper, we propose a variant of the DroQ algorithm, that replaces standard dropout in the Q-functions with advanced dropout. The DroQ variant shows promising results in sparse environments but reaches similar or worse performance in the original environments.

2 Preliminaries

2.1 Maximum Entropy Reinforcement Learning

The standard reinforcement learning procedure is to maximize the expected rewards incurred by an agent learning to act in an environment. The agent interacts with the environment at each time step t, is provided with a state s_t, and responds by selecting an action a_t according to a policy π. After the agent has performed the action, the environment provides a reward r_t and next state s_{t+1} incurred by that action (Hiraoka et al. [11]) .

In maximum entropy reinforcement learning, the agent tries to learn a policy that maximizes the expected rewards with an added entropy bonus as shown in Equation (1).

$$\arg\max_{\pi} \sum_{t=0}^{T} \gamma^T \mathbb{E}_{s_t,a_t} \left[r_t + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right] \tag{1}$$

where γ is the discount factor, α is the temperature parameter that balances exploitation and exploration, and $\mathcal{H}$ is entropy.

2.2 Injecting REDQ with Uncertainty using Dropout Q-Functions

REDQ is a model-free sample-efficient ensemble algorithm for solving maximum entropy RL problems. Its sample efficiency comes from using a high Update-To-Data (UTD) ratio (G), describing the number of agent updates compared to the number of interactions with the environment (Chen et al. [5]). As a consequence of the high UTD ratio, the Q-functions are trained within a few interactions, causing the overestimation bias in the Q-functions to increase. To combat this overestimation, REDQ uses an ensemble of N Q-functions, that helps reduce variance in the Q-function estimate, where only a random subset $\mathcal{M}$ of the Q-functions are minimized (Chen et al. [5]). The large ensemble of Q-functions in REDQ provides great benefits in terms of sample-efficiency and bias reduction but comes at a price of being computationally expensive (Hiraoka et al. [11]).

Using an ensemble of Q-functions in REDQ is a way of injecting uncertainty into the model, which is also achievable by using dropout in the Q-functions. In DroQ, the large ensemble of Q-functions in REDQ is replaced by a smaller ensemble $\mathcal{M}$ of Q-functions with dropout (Srivastava et al. [18]) and layer norm (Ba et al. [3]). DroQ has shown better or similar performance on benchmark MuJoCo environments than REDQ, in addition to being more computationally efficient and memory intensive (Hiraoka et al. [11]).

2.3 Policy-Optimal Information Capacity

Policy Information Capacity (PIC) is the mutual information between policy parameters and episodic return, which is shown to be a well-performing model-agnostic quantitative metric for task complexity (Furuta et al. [7]). In particular, when compared to other alternatives, the PIC variant, Policy-Optimal Information Capacity (POIC), correlates highly with empirical task solvability across a large range of diverse environments. The empirical task solvability in the work by Furuta et al. [7] is approximated using algorithm-based normalized scores (Score(A)) calculated by executing multiple runs of different RL algorithms. POIC is defined as the mutual information between policy parameters Θ and episodic optimality $\mathcal{O}$:

$$\mathcal{I}(\mathcal{O};\Theta) = \mathcal{H}(\mathcal{O}) - \mathbb{E}_{p_{(\theta)}}[\mathcal{H}(\mathcal{O}|\Theta = \theta)] \tag{2}$$

where $\mathcal{H}$ is entropy and $\mathcal{O}$ is the binary random optimality variable explaining whether the agent behaved optimally ($\mathcal{O} = 1$) or not ($\mathcal{O} = 0$) during the entire episode (Furuta et al. [7]). A low POIC or mutual information implies difficult environments, since mutual information is related to the controllability of an environment from the agent's perspective. Intuitively, the less an agent is able to control an environment, the harder it is for the agent to perform well.

3 Extensions of the DroQ Study

3.1 DroQ Performance Across a Larger Range of Environments

In the DroQ and REDQ papers, the algorithms are tested on four MuJoCo benchmark environments: **Ant**, **Hopper**, **Humanoid**, and **Walker2d** (Todorov et al. [19]). In this paper, more environments are explored to examine the existence of patterns in the performance of DroQ depending on the dynamics, difficulty, and reward structure of the task. In addition to the four benchmark environments, this paper includes **HalfCheetah**, **LunarLanderContinuous**, **Pendulum**, and **Reacher**.

Environment	State dim	Action dim	Episode Length	POIC	Score(A)
Pendulum-v1	3	1	200	0.005060	0.710
Ant-v2	111	8	1000	0.000751	0.575
HalfCheetah-v2	17	6	1000	0.000165	0.523
Hopper-v2	11	3	1000	0.000107	0.619
Humanoid-v2	376	17	1000	0.000050	0.466
Reacher-v2	11	2	50	N/A	N/A
Walker2d-v2	17	6	1000	0.000102	0.583
LunarLanderContinuous-v2	8	4	N/A	N/A	N/A

Table 1: Details of environments from Open AI Gym: Classic Control, MuJoCo, and Box2D. POIC and Score(A) are calculated by Furuta et al. [7]. Their work did not include Reacher and LunarLanderContinuous and calculation of these POICs have not been possible given our available computational resources. Additionally, LunarLanderContinuous does not have a fixed episode length, since the termination of an episode is determined by a set of conditions (Brockman et al. [4]).

Table 1 shows information about the state dimension, action dimension, episode length, POIC, and Score(A) of the environments. The POIC and Score(A) both imply that Pendulum is the easiest solvable environment and Humanoid is the most difficult. There are some inconsistencies in the POIC and Score(A), which is evident from their differences in Ant and Hopper.

In addition to introducing a wider range of environments, a sparsity threshold λ is introduced to make environment rewards more sparse and thereby more difficult to learn (Amin et al. [1]). The sparsity threshold increases the difficulty of the task by only providing the agent with a reward once it has crossed a target distance of λ.

3.2 DroQ with Advanced Dropout

Besides exploring how DroQ performs on a larger range of environments, this paper explores whether using advanced dropout in the Q-functions of DroQ can lead to increased performance. Advanced dropout is proposed, as Xie et al. [20] have shown that it leads to a statistically significant accuracy increase compared to other dropout methods, when tested on computer vision data sets, such as MNIST and CIFAR-10. Even though this does not guarantee increased performance in reinforcement learning tasks, advanced dropout uses a model-free approach desirable for any application domain. The model-free methodology is based on a probabilistic graphical model and parametric prior (Xie et al. [20]), which is a stark contrast to many other dropout approaches, where the dropout masks are modeled with model-specific distributions. The dropout masks in advanced dropout are defined as:

$$m_j^{(l)} = \text{Sigmoid}(r_j^{(l)}), \text{ where } r_j^{(l)} \sim \mathcal{N}(\mu_l, \sigma_l^2) \tag{3}$$

where μ_l and σ_l are the mean and standard deviation of the seed variable $r_j^{(l)}$ in the l^{th} layer of a neural network (Xie et al. [20]). In practice, μ_l and σ_l can be approximated by multi-layer perceptrons

as shown in Equation (4) and (5) (Kingma and Welling [15]):

$$\hat{\mu}_l \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{\Omega}_{\mu}^{(l)} \mathbf{h}_i^{(l)} + \mathbf{b}_{\mu}^{(l)} \tag{4}$$

$$\hat{\sigma}_l \approx \frac{1}{N} \sum_{i=1}^{N} \text{Softplus}(\mathbf{\Omega}_{\sigma}^{(l)} \mathbf{h}_i^{(l)} + \mathbf{b}_{\sigma}^{(l)}) \tag{5}$$

$$\hat{\mathbf{h}}^{(l)} \approx \mathbf{\Omega}_{h}^{(l)} \mathbf{x}^{(l)} + \mathbf{b}_{h}^{(l)} \tag{6}$$

where $\mathbf{\Omega}_{\{\mu,\sigma,h\}}^{(l)}$ and $\mathbf{b}_{\{\mu,\sigma,h\}}^{(l)}$ are weights and biases of the multi-layer perceptrons respectively. Advanced dropout has a reduction hyperparameter d controlling the amount of preserved information from the input features in the calculation of $\hat{\mu}_l$ and $\hat{\sigma}_l$. This paper's implementation of advanced dropout is based on the code by Xie et al. [20], where the reduction must be a power of 2. Replacing a standard dropout layer with advanced dropout in a Q-function, increases the number of parameters with $\frac{\text{hidden}_{dim}}{d}(\text{hidden}_{dim} + 2) + 3$ independently of the state and action dimensions of the environment, where hidden_{dim} is the output dimension of a linear layer. This corresponds to a single Q-function having 16.518 more parameters when using $d = 8$ and two hidden layers with a hidden dimension of 256 each. Naturally, DroQ with advanced dropout is more computationally expensive, as more parameters need to be learned. For simplicity, DroQ with advanced dropout is referred to as DroQ$_{AD}$.

4 Experiments

The experiments in this paper evaluate the performance of DroQ and DroQ$_{AD}$ on a larger range of environments as listed in Section 3.1. These are compared with two baseline algorithms: SAC and REDQ. All four algorithms are based on the code from Hiraoka et al. [12], and their hyperparameters for SAC, REDQ, and DroQ are used to conduct the experiments. A high UTD ratio of $G = 20$ is used for all four algorithms and the target entropy of an environment is set according to the MBPO paper by Janner et al. [13]. REDQ uses an ensemble of $N = 10$ Q-functions and updates an ensemble of $|\mathcal{M}| = 2$ Q-functions, which is the size of the smaller ensemble used in SAC, DroQ and DroQ$_{AD}$. The reduction hyperparameter d of $DroQ_{AD}$ is tuned on the Hopper-v2 environment (see Figure 3 in the Appendix). The optimal value of $d = 8$ is used across the remaining seven environments, since hyperparameter tuning for each environment was not possible due to limited computational resources. Using these hyperparameters in Hopper-v2, SAC and DroQ have 141.826 parameters each, DroQ$_{AD}$ has 174.862 parameters, and REDQ has 698.890 parameters.

The algorithms are compared using average return over episodes, as defined in the work by Hiraoka et al. [11], where the average reward across ten episodes is recorded after every epoch. One epoch is considered as 1000 environment interactions for all the environments. The limitations on our computational resources entails that performance is recorded for a single run of every algorithm in an environment.

Figure 1 shows the performance of the algorithms on the eight chosen environments. The DroQ algorithm outperforms the other three algorithms in the HalfCheetah, Humanoid, LunarLanderContinuous and Walker2d environments. These are, with the exception of LunarLanderContinuous, viewed as difficult according to their POIC and Score(A). The algorithms have largely similar performance in the Pendulum and Reacher environments, which are viewed as simpler compared to the rest of the

environments included in this analysis. As opposed to the results by Hiraoka et al. [11], DroQ does not show superior performance in the Ant and Hopper environments. For Ant, REDQ outperforms the rest of the algorithms by far, whereas the difference between the algorithms in Hopper is less significant. This difference is most likely attributed to variations in performance depending on the seed, which are not averaged out in our experiments.

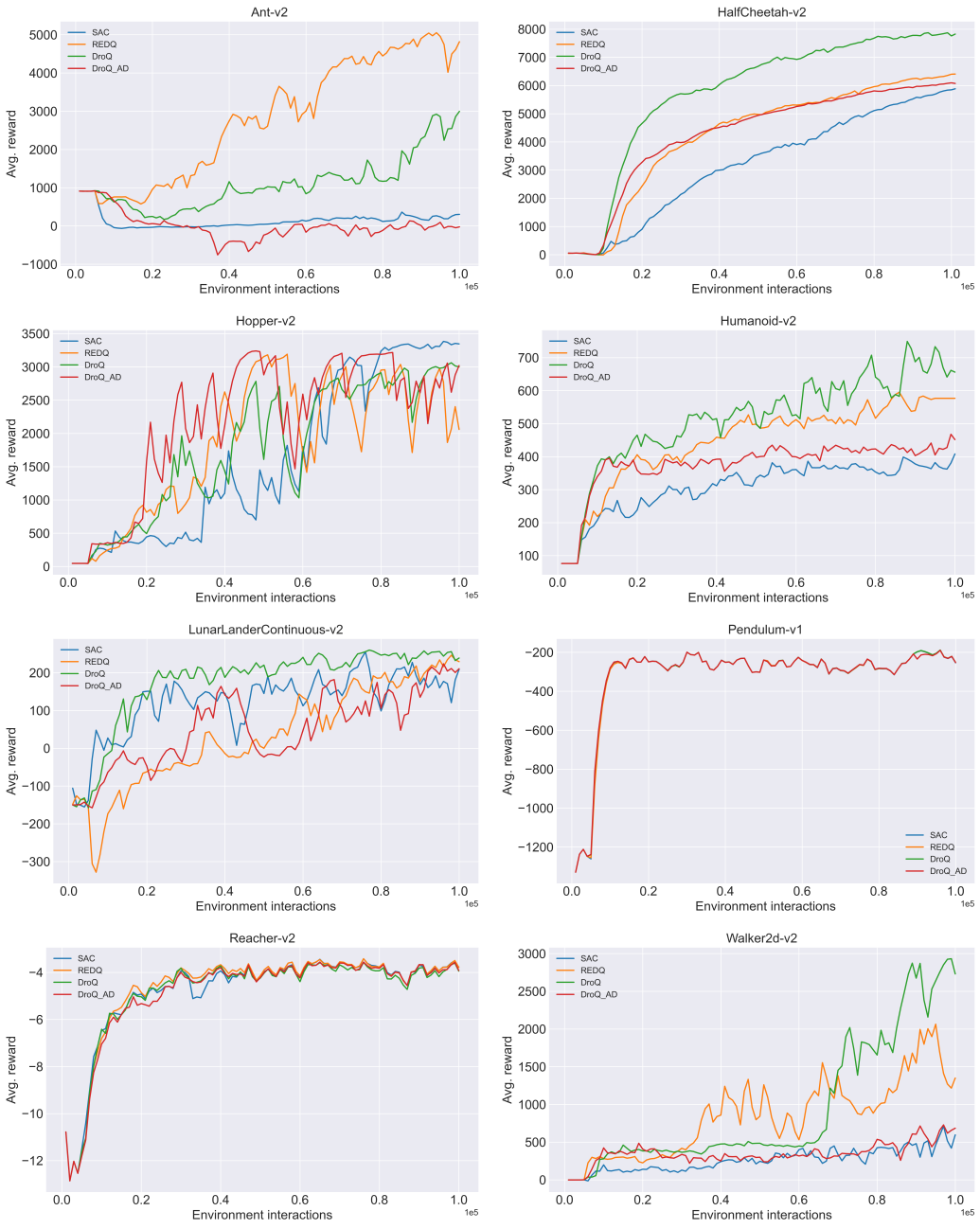

Figure 1: Algorithm performances on wide variety of environments with exponential smoothing using a smoothing factor of $\alpha = 0.4$. Results are from a single run.

The DroQ$_{AD}$ algorithm proposed in this paper, shows subpar performance compared to DroQ in the majority of the environments considered. The best performance is seen on the Hopper environment, in which it was hyperparameter tuned, as it reaches a higher average reward than DroQ.

The performance of the DroQ algorithm does not reveal any patterns when considering POIC, reward structure, episode length, and state and action dimensions of the considered environments. To challenge the algorithms further, the reward structures of Hopper and HalfCheetah are made sparse by using a sparsity threshold of $\lambda = 1$ and $\lambda = 5$ respectively from Amin et al. [1]. The result of this experiment is shown in Figure 2. Surprisingly, the DroQ algorithm flatlines by achieving near zero reward across all interactions, indicating that the small ensemble of dropout Q-functions with layer normalization is not fit for sparse environments. On the other hand, DroQ$_{AD}$ outperforms the other algorithms, despite having a small ensemble like DroQ. In this case, the advanced dropout seems to have a favorable impact on performance, as it is able to adaptively inject uncertainty in a given situation during learning.

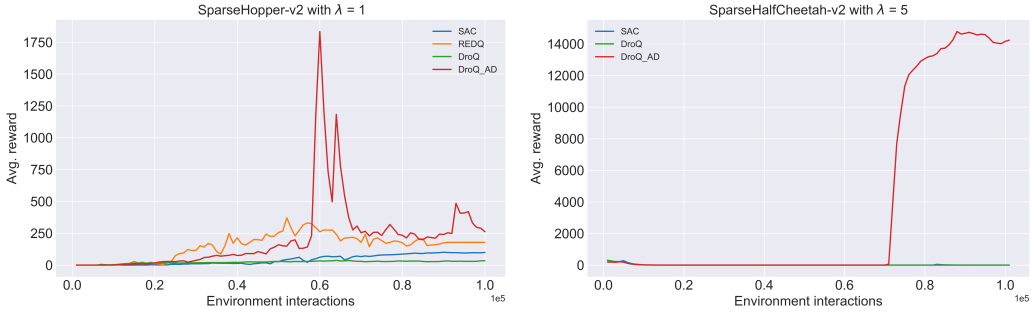

Figure 2: Algorithm performance on sparse versions of Hopper-v2 and HalfCheetah-v2 using $\lambda = 1$ and $\lambda = 5$ respectively and a smoothing factor of $\alpha = 0.4$. Results from Hopper-v2 are an average of two runs and results from HalfCheetah are from a single run.

5 Related Work

To the best of our knowledge, no published studies challenge the performance of the DroQ algorithm on a larger range of environments with sparse reward setups. The key concepts of the DroQ algorithm is not a new area of research, since other papers have explored Q-function ensembles (Faußer and Schwenker [6], He et al. [9]) and dropout used in Q-functions (Jaques et al. [14], Moerland et al. [17]) before its release. However, the methods proposed in previous studies rely on large computationally complex ensembles or dropout in a single Q-function. Prior studies of dropout in Q-functions have been conducted with the standard dropout method, where dropout masks are modeled according to the Bernoulli distribution (Hiraoka et al. [11]), or with Bayesian dropout (Gal and Ghahramani [8]), where the dropout is considered as a Bayesian approximation of uncertainty (Jaques et al. [14], Moerland et al. [17]). We differentiate from previous work by introducing advanced dropout in the Q-functions, exploring whether adaptively modeling the dropout masks improves performance, and by testing the DroQ algorithm on a broader range of environments.

6 Limitations & Future Work

A major limitation of this paper is the lack of computational resources. If computational resources had allowed, multiple runs for each algorithm should have been conducted with different seeds to ensure the robustness of the results and findings in this paper. Additionally, the memory consumption and run times of each algorithm should be recorded to quantify the difference between using advanced dropout in place of standard dropout in DroQ. Furthermore, a larger range of environments could have been included, where the emphasis should be placed on environments that are sparse or considered difficult according to some empirical score (e.g. POIC). A natural addition would be Atari environments with sparse reward structure from Gym (Brockman et al. [4]), such as Montezuma's revenge. Another direction is to have a benchmark environment, where specific changes can be made to the dynamics and reward structure of the environment. This approach could be more helpful to identify the features of an environment that affect the performance of algorithms such as DroQ and DroQ$_{AD}$. Another shortcoming of this paper is that hyperparameter tuning of reduction d for DroQ$_{AD}$ was not conducted on every environment. Consequently, the performance on the other environments could have been improved if this was done.

Future work could use other promising regularization or dropout approaches in the Q-functions of DroQ, that introduce fewer parameters than advanced dropout, such as the adaptive dropout method Standout (Ba and Frey [2]).

7 Conclusion

We conducted an evaluation of DroQ on eight environments from MuJoCo, Box2d, and Classic Control, and found that performance is maintained across the majority of these. Performance in Ant and Hopper was found to be subpar but this could be attributed to the algorithm only being run once. Using DroQ in a sparse reward environment broke performance as it did not achieve any reward during the whole run. We proposed the DroQ$_{AD}$ algorithm as a variant of DroQ, where standard dropout was replaced by advanced dropout in the Q-functions. Our algorithm had superior performance in the sparse reward environment to DroQ, SAC, and REDQ based in two different environments based on two and one runs respectively. However, it showed a subpar performance when evaluated on the eight environments.

Contributions

The majority of this project was conducted in collaboration between the two group members. This includes literature search, building and modifying the code used to conduct experiments, and writing sections in the final paper. Both group members are responsible for all sections in the paper, but the drafts of each section were proposed by the member indicated in Table 2.

Section	Ann-Katrine Christiansen	Jesper Hauch
Abstract	x	x
1. Introduction	x	
2. Preliminaries	2.3	2.1, 2.2
3. Extensions of the DroQ Study	3.1	3.2
4. Experiments	x	x
5. Related Work	x	
6. Limitations & Future Work		x
7. Conclusion		x

Table 2: Project contributions, where 'x' indicates main responsibility for the whole section and numbers indicate main responsibility for subsection(s) within the section.

References

[1] Susan Amin, Maziar Gomrokchi, Hossein Aboutalebi, Harsh Satija, and Doina Precup. Locally persistent exploration in continuous control tasks with sparse rewards, 2020. URL https://arxiv.org/abs/2012.13658.

[2] Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/7b5b23f4aadf9513306bcd59afb6e4c9-Paper.pdf.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

[5] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model, 2021. URL https://arxiv.org/abs/2101.05982.

[6] Stefan Faußer and Friedhelm Schwenker. Neural Network Ensembles in Reinforcement Learning. *Neural Processing Letters*, 41(1):55–69, 2015. ISSN 1573-773X. doi: 10.1007/s11063-013-9334-5. URL https://doi.org/10.1007/s11063-013-9334-5.

[7] Hiroki Furuta, Tatsuya Matsushima, Tadashi Kozuno, Yutaka Matsuo, Sergey Levine, Ofir Nachum, and Shixiang Shane Gu. Policy information capacity: Information-theoretic measure for task complexity in deep reinforcement learning, 2021. URL https://arxiv.org/abs/2103.12726.

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2015. URL https://arxiv.org/abs/1506.02142.

[9] Qiang He, Huangyuan Su, Chen Gong, and Xinwen Hou. Mepg: A minimalist ensemble policy gradient framework for deep reinforcement learning, 2021. URL https://arxiv.org/abs/2109.10552.

[10] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2017. URL https://arxiv.org/abs/1709.06560.

[11] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning, 2021. URL https://arxiv.org/abs/2110.02034.

[12] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xCVJMsPv3RT.

[13] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2019. URL https://arxiv.org/abs/1906.08253.

[14] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR*, abs/1907.00456, 2019. URL http://arxiv.org/abs/1907.00456.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

[16] Marian Körber, Johann Lange, Stephan Rediske, Simon Steinmann, and Roland Glück. Comparing popular simulation environments in the scope of robotics and reinforcement learning, 2021. URL https://arxiv.org/abs/2103.04616.

[17] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Efficient exploration with double uncertain value networks, 2017. URL https://arxiv.org/abs/1711.10789.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[19] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[20] Jiyang Xie, Zhanyu Ma, Jianjun Lei, Guoqiang Zhang, Jing-Hao Xue, Zheng-Hua Tan, and Jun Guo. Advanced dropout: A model-free methodology for bayesian dropout optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/tpami.2021.3083089. URL https://doi.org/10.1109%2Ftpami.2021.3083089.

[21] Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning, 2018. URL https://arxiv.org/abs/1811.06032.

Appendix

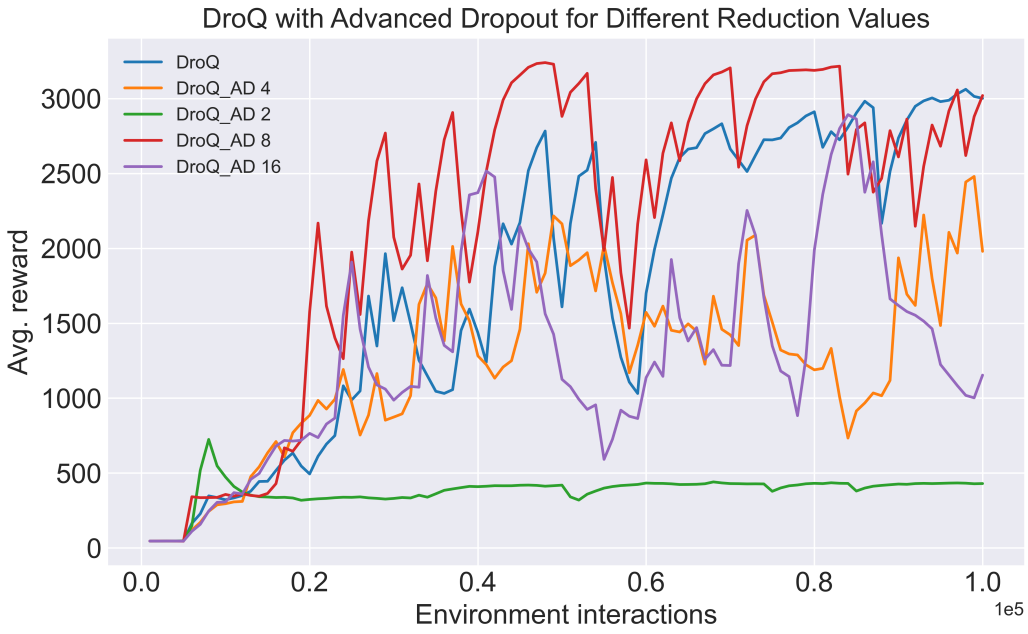

Figure 3: Hyperparameter tuning of reduction parameter λ in advanced DroQ for environment Hopper-v2. A reduction of $d = 8$ was selected and used for the remaining environments.