Bayesian Exploration in Deep Reinforcement Learning

Abstract

1 INTRODUCTION

A reinforcement learning environement is modelled as a Markov decision process (MDP) $M = \langle \mathcal{S}, \mathcal{A}, r, P, P_0, \gamma \rangle$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the set of available actions. At time $t = 0$ a state s_0 is sampled from the distribution $P_0(\cdot)$. At each timestep an action $a_t \sim \pi(\cdot|s_t)$ is selected and the agent transitions to a new state state $s_{t+1} \sim P(\cdot|s_t, a_t)$. A scalar reward $r_t = r(\cdot|s_t, a_t)$ is observed. As the agent and environement interact in a sequence of time steps, a history of observations $\mathcal{H}_t = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_t, a_t, r_t)$ is collected. The goal is to find a policy $\pi^\star$, such that sampling actions $a \sim \pi^*(\cdot|s)$ maximizes the expected accumulated and discounted future reward, $J^\pi := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. An efficient agent must be able to learn from the data it collects, but since the data is dependent on the policy, it must also prioritize to explore states and actions that the agent can learn a lot from.

The efficiency of exploration can be measured in regret. The regret of a policy is the difference in the expected reward obtained by following that policy, and the expected reward of following an optimal policy. An learning algorithm's efficiency in exploration can be measured by its cumulative regret over time. One of the simplest reinforcement learning problems is the multi-arm bandit problem. This is an MDP with no state and no transition probabilities. One exploration strategy employed in most Q-learning algorithms to date, ϵ-greedy is provably inefficient, and has a regret bound that grows linearly with time. There are several optimal algorithms for this problem, but perhaps the simplest one is Thompson sampling [Thompson, 1933]. Thompson sampling approximates a posterior distribution of the expected reward for each action. The next action is decided by sampling rewards for each action from the posterior distribution, and selecting the action that gave the highest reward. Bayesian methods has also been shown to behave efficiently with respect to cumulative regret Osband et al. [2016] on general MDPs.

The Q-function, $Q^\pi(s, a)$, is defined as the expected reward of taking action a in state s and then following policy π thereafter: $Q^\pi(s_0, a_0) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = S, a_0 = A\right]$. The Bellman operator on Q^π is defined as $\mathcal{B}[Q^\pi](s_t, a_t) := \mathbb{E}_{P(s_{t+1}|s_t,a_t)\pi(a_{t+1}|s_t)} \left[r(s_t, a) + \gamma Q^\pi(s_{t+1}, a_{t+1})\right]$. With the Q-function we can define a policy that always picks the action with the highest Q-value in any state. In deep RL, we model Q with a neural network $\hat{Q}_\omega$. One way to facilitate exploration in this policy is to introduce uncertainty in the Q-value function.

Fortunato et al. [2019] introduced NoisyNet for exploration. These are networks with stochastic weights, where each weight $w_{ij}^{(l)}$ has an added pertrubaition sampled from a noise distribution with standard deviation $\sigma_{ij}^{(l)}$. After initializing $\boldsymbol{w}$ and $\boldsymbol{\sigma}$ such that the network has sufficient stochasticity for exploration, both parameters are learned using standard backpropagation. The approach is similar to variational inference schemes such as Bayes by backprop [Blundell et al., 2015] where the weights of a neural network model are assumed to be normally distributed with mean $\mu_{ij}^{(l)}$ and standard deviation $\sigma_{ij}^{(l)}$. The objective of Bayes by backprop, however, is to approximate the posterior distribution $p(\boldsymbol{w}|\mathcal{D})$ in a task with a fixed dataset $\mathcal{D}$. The parameter distribution in NoisyNet does not necessarily converge to an approximate posterior. This means that it does not have the same guarantee on total regret as methods that approximate a posterior over the value functions [Osband et al., 2016]. They do, however, have experimental results which shows that the function does not always converge to a deterministic solution, but it is unclear why the network learns to introduce more noise into the network parameters.

Fortunato et al. [2019] apply NoisyNet to three reinforcement learning algorithms, DQN, Dueling DQN, and A3C, and show improved performance on all of them. Later NoisyNet was used in the Rainbow algorithm [Hessel et al., 2017], a combination of six extensions to the DQN algorithms [Fortunato et al., 2019, Bellemare et al., 2017, Wang et al., 2016, van Hasselt et al., 2015, Schaul et al., 2016, Mnih et al., 2016], that shows state-of-the-art performance across 57 Atari games.

In this paper, we will introduce an extension to NoisyNet that puts it into a Bayesian context. A limitation of NoisyNet is that the initial uncertainty in the Q-value function is crucial to exploration. If the uncertainty is too high, algorithm will struggle to learn anything, while if the uncertainty is too low, there is nothing incentivizing exploration, and the algorithm will likely be stuck in a poor local minimum.

2 BACKGROUND

2.1 NOISY NETWORKS

In NoisyNet [Fortunato et al., 2019], the Q-function space can be seen as a stochastic process on the probability space $(\Omega, \mathcal{F}, P)$. The stochastic process can simply be written as $\{Q(\boldsymbol{s}, \boldsymbol{a}) : (\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S} \times \mathcal{A}\}$. For any $\omega \in \Omega$, $Q(\cdot, \cdot, \omega)$ is a sample function mapping $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. To simplify notation we will denote sample functions as f. NoisyNet, specifically, is a stochastic process parameterized by its weights and their standard deviation. We will define this parameterized function as g_ϕ, and their sampling distribution as $\rho_\phi \in \mathcal{P}$. Given a noise vector ξ, we have $f(\boldsymbol{s}, \boldsymbol{a}) = g_\phi(\boldsymbol{a}, \boldsymbol{s}, \xi)$. This means that NoisyNet gives a new policy function for each noise vector ξ. Although this means that NoisyNet can model stochastic policies, Fortunato et al. [2019] point out that for the loss function it is optimising, there always exsists a deterministic optimal policy. They show through emipirical analysis, however, that this does not mean that the policy necessarily disregards the noise and converges to a deterministic policy.

2.2 FUNCTIONAL VARIATIONAL BAYESIAN NEURAL NETWORKS

There are several ways of approximating the posterior distribution of the weights in a neural network with respect to a prior and a dataset. Typically, the dataset $\mathcal{D}$ is static and with datapoints $\boldsymbol{x}$ and labels $\boldsymbol{y}$, and the prior is defined as a distribution over the weights $p(\boldsymbol{w})$ [Rezende et al., 2014, Blundell et al., 2015, Ritter et al., 2018, Maddox et al., 2019]. By defining a prior over the weights, they can use approximate variational inference methods to approximate $p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})}$. The disadvantage of this is that $p(\boldsymbol{w})$ only acts as a regularizer, and is not used to incorporate prior knowledge. Any prior knowledge we might have about the optimizer function would be very difficult to translate into a prior distribution in the weight space.

Functional variational Bayesian neural networks [Sun et al., 2019] is a variational inference method for neural networks that approximates the posterior distribution in function space. This means that our prior will be a distribution over functions, also known as a stochastic process. Sun et al. [2019] show that for two stochastic processes A and B, the KL-divergence from A to N is the supremum of the marginal KL-divergences over all finite measurement sets. Let $A_{\mathbf{X}}$ be the marginal distribution of function values at $\mathbf{X} \in \mathcal{X}^n$, then:

$$\mathrm{KL}[A \| B] = \sup_{n \in \mathbb{N}, \mathbf{X} \in \mathcal{X}^n} \mathrm{KL}\left[A_{\mathbf{X}} \| B_{\mathbf{X}}\right], \qquad (1)$$

Now the stochastic processes is represented by a neural network ρ_ϕ and a prior p. In a similar manner to NoisyNet, we sample a function f from ρ_ϕ by sampling a random noise vector ξ, and then defining $f(\boldsymbol{x}) = g_\phi(\boldsymbol{x}, \xi)$. They further show that the gradient of the KL-divergence for functions evaluated at the measurement set $\mathbf{X}$ is

$$\begin{aligned} \nabla_\phi \mathrm{KL}[q_\phi(\mathbf{f}(\mathbf{X})) \| p(\mathbf{f}(\mathbf{X}))] &= \mathbb{E}_\rho\left[\nabla_\phi \log \rho_\phi(\mathbf{f}(\mathbf{X}))\right] \\ &+ \mathbb{E}_\xi\left[\nabla_\phi \mathbf{f}(\mathbf{X})(\nabla_{\mathbf{f}} \log \rho_\phi(\mathbf{f}(\mathbf{X})) - \nabla_{\mathbf{f}} \log p(\mathbf{f}(\mathbf{X})))\right]. \end{aligned} \qquad (2)$$

The measurement set $\mathbf{X}$ is created by concatenating the training points $\mathbf{X}^D$ and a set of M points $\mathbf{X}^M$ drawn from some distribution c that has full support in the space of interesting test cases.

The first term is just the expected value of the score function, which is zero. The difficult part in (2) is to estimate $\nabla_{\mathbf{f}} \log \rho(\mathbf{f}^{\mathbf{X}})$ and $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$. $\nabla_{\mathbf{f}} \log \rho(\mathbf{f}^{\mathbf{X}})$ is likely intractable, considering ρ_ϕ is a neural network with stochastic weights. Depending on how we define the prior, however, $\nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}})$ can be easy to compute analytically. To reduce variance in the gradients, we define tractable priors in this paper. To estimate the log-density gradient $\nabla_{\mathbf{f}} \log \rho(\mathbf{f}^{\mathbf{X}})$, they use the Spectral Stein Gradient Estimator (SSGE) [Shi et al., 2018]. In short, this is a method for estimating the gradient function of implicit distributions using approximations to eigenfunctions of a kernel-based operator. We bring three hyperparameters into the algorithm from this method, the number of samples k from the implicit distribution used to approximate the gradient, the number of eigenvectors J used to approximate the gradient, and η, a regularization parameter that smooths the gradient function.

The full minimization target for the functional Bayesian neural network becomes

$$\begin{aligned} &\frac{1}{|\mathcal{D}_s|} \sum_{(\boldsymbol{x}, y) \in D_s} \mathbb{E}_{\rho_\phi}\left[p(y | f(\boldsymbol{x}))\right] \\ &\quad - \lambda \mathrm{KL}\left[\rho_\phi\left(\mathbf{f}(\mathcal{D}_s), \mathbf{f}(\mathbf{M})\right) \| p\left(\mathbf{f}(\mathcal{D}_s), \mathbf{f}(\mathbf{M})\right)\right]. \end{aligned} \qquad (3)$$

In order for this to match the functional ELBO and be a proper lower bound for $\log p(\mathcal{D})$, λ should be set as $\frac{1}{|\mathcal{D}|}$. Sun et al. [2019] note, however, that the minimization target is only a lower bound for KL $[|\rho(f)\|p(f)]$, so a larger value for λ is likely necessary to prevent overfitting. They use, $\lambda = \frac{1}{|\mathcal{D}_s|}$, a value of one over the batch size instead. This means that the minimization target becomes a proper lower bound of $\log p(\mathcal{D}_s)$.

3 METHOD

Osband and Van Roy [2017] prove that in finite horizon MDPs, posterior sampling for reinforcement learning has at least a near-optimal regret bound. They further conjecture that this can be improved to show an optimal regret bound. Additionally, a posterior sampling-based reinforcement learning algorithm can be made to utilize prior information or domain knowledge through an appropriate prior. This should improve the policy convergence rate. Combined with the computational efficiency of posterior sampling, this motivates the development of a Bayesian reinforcement learning algorithm with functional priors.

We will present a method based on functional variational Bayesian neural networks [Sun et al., 2019] that allows the following:

- Efficient exploration
- Incorporate domain knowledge
- Avoid hyperparameter optimization

We would like to approximate the posterior distribution of the Q-value function.

We can use the functional variational Bayesian neural network (FVBNN) [Sun et al., 2019] framework discussed earlier. NoisyNet uses one sample from Q for each optimization step. We will instead use k samples from Q. This lets us add the KL-term from FVBNN to the loss function that encourages stochasticity. The maximization target becomes:

$$\log p(\mathcal{D}_\omega^N|f) - \lambda \mathrm{KL}\left[q(\mathbf{f}^{\mathcal{D}_\omega^N}, \mathbf{f}^{\mathbf{M}}) \| p(\mathbf{f}^{\mathcal{D}_\omega^N}, \mathbf{f}^{\mathbf{M}})\right], \quad (4)$$

where $\mathbf{f}^{\mathcal{D}_\omega^N}$ is f applied to the dataset $\mathcal{D}_\omega^N$, and $\mathbf{f}^{\mathbf{M}}$ is f applied to a set of i.i.d. random points $\mathbf{M} = \{m_i \sim c \mid i = 1, \dots, k\} \subseteq \mathcal{S} \times \mathcal{A}$ with full support. We will let the measurement $\mathbf{M}$ be the the set of all state-action pairs for states we have already explored. If the MDP is ergodic, then $\mathbf{M}$ will eventually have full support in $\mathcal{S} \times \mathcal{A}$.

$$\mathbf{M} = \{(s_i, a_j)\ \forall s_i \in \mathcal{D}_\omega^N,\ \forall a_j \in \mathcal{A}\}. \quad (5)$$

To calculate $\log p(\mathcal{D}_\omega^N|f)$ we assume that the observation noise is sampled from a Gaussian distribution. We then model the network to produce two output vectors, μ and $\tau = \log \sigma$. This gives the following loss function:

$$\mathcal{L} = \frac{1}{|\mathcal{D}_s|} \sum_{i=1}^{|\mathcal{D}_s|} \tau_i + (\mu_i - q_i)^2 \exp(-\tau_i) + \lambda \mathrm{KL}[\rho \| p]. \quad (6)$$

We take inspiration from Sun et al. [2019] and let λ be one over the batch size, $\frac{1}{|\mathcal{D}_s|}$.

Algorithm 1 DQN-update

```
$\mathcal{D} \leftarrow \emptyset$
$s \sim P_0$
Initialize $\phi, \omega$
while not converged do
    $f \sim q_\phi$
    $a \sim \arg\max_a f(s, a)$
    $s' \sim P(\cdot|s, a)$
    $r = r(s', a, s)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{s, a, r, s'\}$
    UPDATENET($\phi, \omega, \mathcal{D}$)
end while
```

Algorithm 2 NoisyNet RL

```
function UPDATENOISYNET($\phi, \omega, \mathcal{D}$)
    $T \sim \mathcal{D}$
    $f \sim q_\phi$
    $b \sim q_\omega$
    $\Delta_{\mathcal{L}} \leftarrow 0$
    for $\{s, a, r, s'\} \in T$ do
        $q \leftarrow \max_a f(s, a)$
        $G \leftarrow r + \gamma b(s, a)$
        $\Delta_{\mathcal{L}} \leftarrow \Delta_{\mathcal{L}} - \frac{1}{|T|} \nabla_\phi \log p(q|G)$
    end for
    $\phi \leftarrow \phi + \eta_\phi \Delta_{\mathcal{L}}$
    $\omega \leftarrow \omega + \eta_\omega (\omega - \phi)$
end function
```

4 EXPERIMENTS

Our method most closely resembles that of NoisyNet Fortunato et al. [2019], so all experiments will mainly compare the performance of these two methods.

4.1 DETAILS AND HYPERPARAMETERS

We compare our method with NoisyNet [Fortunato et al., 2019] on four different environments in the OpenAI Gym [Brockman et al., 2016b]. The hyperparameters are the same as those described in Han et al. [2020]. Our method additionally has hyperparameters related to the functional KL-divergence. These are reported in Table 1.

Algorithm 3 Functional Bayesian RL

```
function UPDATEPOSTERIOR($\phi, \omega, \mathcal{D}$)
    $T \sim \mathcal{D}$
    $\mathbf{f} \leftarrow \{f_i \sim q_\phi \mid i = 1, \dots, k\}$
    $\mathbf{b} \leftarrow \{b_i \sim q_\omega \mid i = 1, \dots, k\}$
    $\mathcal{F} \leftarrow \emptyset$
    $\Delta_{\mathcal{L}} \leftarrow 0$
    $S \leftarrow 0$
    for $\{s, a, r, s'\} \in T$ do
        $\boldsymbol{q_\mu}, \boldsymbol{q_\sigma} \leftarrow \max_a \mathbf{f}(s, a)$
        $\mathcal{F} \leftarrow \mathcal{F} \cup \mathbf{f}(s, a)$
        $\boldsymbol{G} \leftarrow \{r + \gamma b_i(s, a) \mid i = 1, \dots, k\}$
        $S \leftarrow S + \frac{1}{k} p(\boldsymbol{q}|s, a)$
        $\Delta_{\mathcal{L}} \leftarrow \Delta_{\mathcal{L}} - \frac{1}{|k|} \nabla_\phi \log p(\boldsymbol{q_\mu}|\boldsymbol{G}, \boldsymbol{q_\sigma})$
    end for
    $\Delta_{\mathrm{KL}} \leftarrow S - SSGE(\mathcal{F})$
    $\phi \leftarrow \phi + \eta_\phi(\Delta_{\mathcal{L}} - \Delta_{\mathrm{KL}})$
    $\omega \leftarrow \omega + \eta_\omega(\omega - \phi)$
end function
```

Table 1: Hyperparameters for LFM.

	Hyperparameter	Value
k	Number of function samples	100
J	Number of eigenvectors	6
λ	KL regularization parameter	$\frac{1}{\lvert\mathcal{D}_s\rvert}$
η	Regularization coefficient for SSGE	0.95

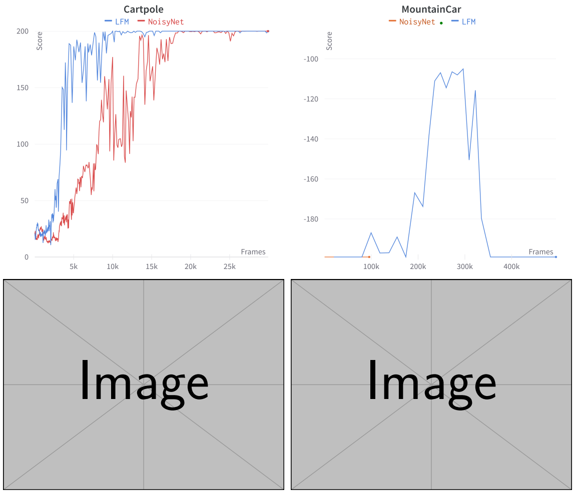

Figure 1: Comparison of learning curves of LFM and NoisyNet-DQN on Pong, Cartpole, MountainCar, and Acrobot averaged over 100 episodes.

4.2 UNINFORMATIVE PRIOR

First, it is interesting to see how our method will compare to NoisyNet in environments without the advantage of prior knowledge. The training curves of our model compared to that of standard NoisyNet-DQN on the four selected OpenAI Gym [Brockman et al., 2016a] environments is shown in Figure 1. We hypothesize that the added KL-term will push the algorithm to retain weight uncertainty high in order to keep a higher uncertainty in the output. While both methods can find an optimal policy in Cartpole, our method uses considerably fewer frames to do so.

Future

Fortunato et al. [2019] noted that the learned variance in their weight increased in some games despite there existing an optimal deterministic solution, and the loss provides no incentive to maintain any uncertainty. Figure 2 shows the mean-absolute standard deviation for the penultimate and final layer. It is interesting to see that the standard deviation for NoisyNet continues to decrease throughout the training while LFM decreases faster, but then flattens out earlier. This seems to indicate that it has found a stable policy, where optimizing further would be overfitting to noise.

4.3 INFORMATIVE PRIOR

One of the benefits of a functional prior is that we can incorporate domain knowledge to get more efficient exploration. We will now look at how our method can utilize domain knowledge to improve sample efficiency.

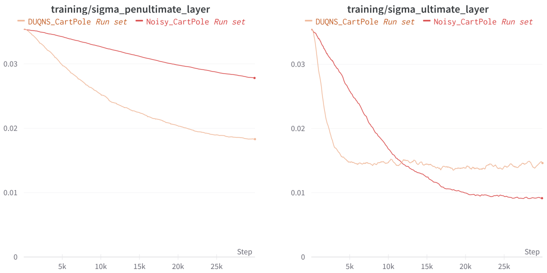

Figure 2: Comparison of mean-absolution standard deviation for NoisyNet and LFM on Cartpole.

5 CONCLUSION AND DISCUSSION

We have presented a method for exploration with a DQN-style algorithm that can effectively utilize domain knowledge for faster learning. Our results show that it outperforms standard DQN with NoisyNet exploration with regards to sample efficiency on all four selected environments even without prior knowledge.

One interesting avenue for future work is to extend the approach to methods other than the standard DQN. A functional Bayesian approach for policy evaluation would permit direct prior distribution in policy space rather than in value space, which seems like a more intuitive prior in many domains.

References

Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. *arXiv:1707.06887 [cs, stat]*, July 2017. URL http://arxiv.org/abs/1707.06887. arXiv: 1707.06887.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, July 2015. PMLR. URL http://proceedings.mlr.press/v37/blundell15.html.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016a. URL http://arxiv.org/abs/1606.01540. arXiv: 1606.01540.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016b. URL http://arxiv.org/abs/1606.01540. arXiv: 1606.01540.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks for Exploration. *arXiv:1706.10295 [cs, stat]*, July 2019. URL http://arxiv.org/abs/1706.10295. arXiv: 1706.10295.

Shuai Han, Wenbo Zhou, Jing Liu, and Shuai Lü. NROWAN-DQN: A Stable Noisy Network with Noise Reduction and Online Weight Adjustment for Exploration. *arXiv:2006.10980 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2006.10980. arXiv: 2006.10980.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. *arXiv:1710.02298 [cs]*, October 2017. URL http://arxiv.org/abs/1710.02298. arXiv: 1710.02298.

Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. February 2019. URL https://arxiv.org/abs/1902.02476v2.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*, June 2016. URL http://arxiv.org/abs/1602.01783. arXiv: 1602.01783.

Ian Osband and Benjamin Van Roy. Why is Posterior Sampling Better than Optimism for Reinforcement Learning? *arXiv:1607.00215 [cs, stat]*, June 2017. URL http://arxiv.org/abs/1607.00215. arXiv: 1607.00215.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and Exploration via Randomized Value Functions. *arXiv:1402.0635 [cs, stat]*, February 2016. URL http://arxiv.org/abs/1402.0635. arXiv: 1402.0635.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv:1401.4082 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1401.4082. arXiv: 1401.4082.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural Networks. February 2018. URL https://openreview.net/forum?id=Skdvd2xAZ.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. *arXiv:1511.05952 [cs]*, February 2016. URL http://arxiv.org/abs/1511.05952. arXiv: 1511.05952.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. A Spectral Approach to Gradient Estimation for Implicit Distributions. *arXiv:1806.02925 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1806.02925. arXiv: 1806.02925.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. *arXiv:1903.05779 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1903.05779. arXiv: 1903.05779.

William R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25 (3/4):285–294, 1933. ISSN 0006-3444. doi: 10.2307/2332286. URL https://www.jstor.org/stable/2332286. Publisher: [Oxford University Press, Biometrika Trust].

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. *arXiv:1509.06461 [cs]*, December 2015. URL http://arxiv.org/abs/1509.06461. arXiv: 1509.06461.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. *arXiv:1511.06581 [cs]*, April 2016. URL http://arxiv.org/abs/1511.06581. arXiv: 1511.06581.