MEET: A MONTE CARLO EXPLORATION-EXPLOITATION TRADE-OFF FOR BUFFER SAMPLING

*Julius Ott[1,2] *Lorenzo Servadei[1,2] Jose Arjona-Medina[3] Enrico Rinaldi[4]
Gianfranco Mauro[1] Daniela Sánchez Lopera[1,2] Michael Stephan[1]
Thomas Stadelmayer[1] Avik Santra[1] Robert Wille[2]

[1]Infineon Technologies AG, [2] Technical University of Munich,
[3]Johannes Kepler University Linz, [4]University of Michigan

ABSTRACT

Data selection is essential for any data-based optimization technique, such as Reinforcement Learning. State-of-the-art sampling strategies for the experience replay buffer improve the performance of the Reinforcement Learning agent. However, they do not incorporate uncertainty in the Q-Value estimation. Consequently, they cannot adapt the sampling strategies, including exploration and exploitation of transitions, to the complexity of the task. To address this, this paper proposes a new sampling strategy that leverages the exploration-exploitation trade-off. This is enabled by the uncertainty estimation of the Q-Value function, which guides the sampling to explore more significant transitions and, thus, learn a more efficient policy. Experiments on classical control environments demonstrate stable results across various environments. They show that the proposed method outperforms state-of-the-art sampling strategies for dense rewards w.r.t. convergence and peak performance by 26% on average.

Index Terms— uncertainty estimation, experience replay, reinforcement learning

1. INTRODUCTION

In *Deep Reinforcement Learning (DRL)* applications, the buffer, where experiences are saved, represents a key component. In fact, learning from stored experiences leverages supervised learning techniques, in which Deep Learning excels [1]. Seminal work has shown how buffer sampling techniques improve the performance of DRL models over distributions observed during training [2]. Consequently, how to sample from the buffer plays an important role in the learning process. In this context, a major component of the buffer sampling strategy regards the uncertainty of the agent in choosing the optimal action. This influences the trade-off between exploration-exploitation in the buffer sampling strategy.
In the literature, the concept of uncertainty has been applied to tasks performed by a *Machine Learning (ML)* model over unseen data distributions. Those are called *Out-of-Distribution (OOD)* data, i.e., samples for which the model has high uncertainty. Thus, in the state of the art, the assessment of that uncertainty is typically used for OOD detection. For instance, [3] proposes an uncertainty-based OOD-classification framework called UBOOD, which uses the epistemic uncertainty of the agent's value function to classify OOD samples. In particular, UBOOD compares two uncertainty estimation methods: dropout- and bootstrap-based. The highest performance is achieved using bootstrap-based estimators, which leverage the bootstrap neural network (BootDQN) [4].
Inspired by [3, 4], this paper employs a bootstrap mechanism with multiple heads for determining the uncertainty in the Q-Value estimation. This is exploited by the proposed novel algorithm: a **M**onte Carlo **E**xploration- **E**xploitation **T**rade-Off (MEET) for buffer sampling. Thanks to the Q-value uncertainty estimation, MEET enables an optimized selection of the transitions for training Off-Policy *Reinforcement Learning (RL)* algorithms and maximizes their return. We evaluate MEET on continuous control problems provided by the MuJoCo [1] physics simulation engine. Results show that MEET performs consistently in terms of convergence speed and improves the performance by 26% in challenging, continuous control environments.
The remainder of this paper is structured as follows: in Section 2, we present the background related to continuous RL, buffer sampling, and uncertainty estimation. Furthermore, we motivate the necessity of the proposed approach. In Section 3, we introduce the proposed buffer sampling strategy, while Section 4 describes the performed experiments on public datasets and the obtained results. Finally, Section 5 concludes the paper.

2. BACKGROUND AND RELATED WORK

In this section, we present concepts related to the approach introduced in this paper. To this end, we first present characteristics of continuous RL, and then we review the role of uncertainty in RL.

2.1. Continuous Reinforcement Learning

Traditional RL methods often assume a finite action space. In real-world applications, however, RL methods do face a continuous action space. Different methods have been developed to extend existing methods to continuous action spaces. The *Deterministic Policy Gradient (DPG)* method is one prominent example [5]. The parameters are changed in the direction of the policy gradient using the same method as for stochastic

*Equal contribution

[1]`github.com/deepmind/mujoco`

policies. In this method, exploration is achieved by learning from samples uncorrelated to the current policy. This work was extended in Deep Deterministic Policy Gradient [6], by combining Deep Q-Learning and DPG, at the expense of losing the convergence guarantees of DPG. As an alternative to the policy gradient method, a continuous variant of the Q-Learning algorithm (Normalized Advantage Functions) enables the use of Q-Learning in continuous action spaces. In this method, the advantage function is parameterized as a quadratic function of nonlinear features of the state [7]. In this way, the action that maximizes the Q-Value function can be determined analytically during the Q-Learning update. Soft Actor-Critic methods [8] combine Off-Policy training with a stochastic actor, which also maximizes the entropy. As a result, training stability increases along with efficiency and performance.

In desirable scenarios, optimal control is reached by having a model of the environment. Nevertheless, in many real-world applications, a model of the environment cannot be accurately estimated. Therefore, in those cases, model-free DRL is often used. Model-free On-Policy learning is known to be very sample inefficient, as data is not reused [9]. Sampling trajectories from the environment can be costly and slow. To overcome this, the RL algorithm could rely on additional trajectories, generated by other policies, which can be used in the learning phase. As a matter of fact, Off-Policy methods can deal with trajectories sampled from different policies, improving the sample efficiency and exploration. One of the key elements for the success of DQN [10] was the usage of a replay buffer or experience replay [11], where transition tuples (state, action, next-state, and reward) are stored. This buffer then serves as a training dataset from which supervised learning techniques can be applied. A naive approach would sample data evenly. However, there might be better strategies to sample from the buffer. For example, when transition tuples are stored to learn a Q-Value function using Q-Learning, biased sampling towards high *Temporal Difference (TD)* error tuples can speed up the learning time [12]. The same idea has been applied to trajectories as well [13]. When rewards are sparse or delayed, storing trajectories with high returns and sampling them more often also speeds up learning [14]. A different approach compares states along new trajectories with the states in the buffer to filter out those that are novel [15]. Even RL has been used to learn an auxiliary policy that samples from the replay buffer the most useful experiences [16].

Uncertainty estimation has been used already in off-line RL [17], trajectory exploration [18] safe RL [19], and for sampling strategies in discrete [2] as well as continuous action spaces [20]. However, to the best of our knowledge, the uncertainty of the Q-Value estimation has not yet been used to leverage exploration and exploitation for buffer sampling. In that way, the proposed method samples more useful transitions in continuous action space problems, without any further assumptions on the RL model.

2.2. Uncertainty-Based Reinforcement Learning

Uncertainty-based RL aims to provide a policy with a corresponding policy estimate. A When training data comes from a different distribution, *Deep Neural Network (DNN)* trained with supervised learning techniques may underperform in tests [21]. Methods for quantifying uncertainty in predictions have recently been focused on image and text classification tasks [22]. In RL, uncertainty-based methods have been used for OOD detection. In particular, these methods focus on the DQN. The goal is to estimate how certain the agent is to choose an action as optimal. A DQN update is determined by the current state s_t, action a_t, reward r_t, and next state s_{t+1} in the form of $\theta_{t+1} = \theta_t + \eta(y_t^Q - Q(s_t, a_t; \theta_t))\nabla_\theta Q(s_t, a_t; \theta)$, where θ are the network parameters, y_t^Q the target value at episode step t, and η, the learning rate. To estimate the uncertainty of the Q-Value function approximation, three methods are compared in [3]: Monte-Carlo Concrete Dropout [23], and Bootstrap methods leveraging BootDQN [4] with and without a random prior Network [24]. Monte Carlo Concrete Dropout employs dropout layers, which learn individual dropout rates per layer. This avoids hyperparameter search for optimal rates and couples with changing data during training, which happens in an RL setting.

Bootstrap-based methods outperform Monte Carlo Concrete Dropout by leveraging the statistical idea of bootstrapping, i.e., approximating a population by a sample distribution using a model ensemble. The bootstrapped DQN [4] approximates a "distribution" of Q-Values, not to be confused with distributional RL. It trains L estimates of the Q-Value function $\hat{Q}_l(s, a, \theta)$ against its target network $\hat{Q}_l(s, a, \theta^-)$. Two architectures are possible: an ensemble of L DNNs estimating $\hat{Q}_l$-value functions or one DNN with L heads. The multi-head strategy is more efficient and utilizes the same memory buffer over L estimations without requiring parallelization. The multi-head bootstrapped DQN modifies DQN by adding L heads or L Q-Value functions. For each episode step t, a value function $\hat{Q}_l$ is selected to act by choosing $l \in \{1, ..., L\}$ from a uniform and random distribution. For each step of the episode, the action a maximizing $\hat{Q}_l(s_t, a)$ is executed. Afterward, a masking probability distribution M generates m_t, a mask to identify whether the experience in t should be used for training. The current transition tuple and the mask m_t^l are stored in the replay buffer, which is common to all L heads. The gradients g of the value function $\hat{Q}_l$ in a time step t are $g_t^l = m_t^l(y_t^Q - \hat{Q}_l(s_t, a_t; \theta))\nabla_\theta \hat{Q}_l(s_t, a_t; \theta)$. DQN refers to an only-critic algorithm, referring to [25]. Thus, we can extend the idea of bootstrapped DQN to the Actor-Critic method without loss of generality.

3. CONSTRUCTING A BUFFER SAMPLING STRATEGY BASED ON UNCERTAINTY

In this section, we introduce MEET, an algorithm that leverages both exploration and exploitation for an improved buffer sampling strategy. To this end, we first review how the proposed method relates to *Upper Confidence Bound (UCB)* algorithms. Afterward, we explain in detail the implementation of MEET.

3.1. A Sampling Strategy for Exploration-Exploitation Trade-off

Many techniques target the exploration-exploitation dilemma in RL. Although those techniques have been applied to reward functions, as well as to states and actions selection, they have not yet been studied for transition sampling in buffers. Accordingly, we imagine the transition sampling as a decision-making problem, similar to the UCB algorithm in the Multi-Armed Bandit problem. Here in fact the decision-making happens, under uncertainty, by selecting one of k-armed bandits at each time step. A decision-maker or agent is present in the Multi-Armed Bandit Problem to choose between k-different actions and receives a reward based on the selected action. In this paper, we consider UCB1, which trades-off exploitation and exploration, and is formalized as:

$$\text{UCB1}(i) = \mu_i + \sqrt{\frac{2\ln{(N)}}{N_i}}, \quad (1)$$

where μ_i represents the current reward average of arm i at the current round; N the number of trials passed; and N_i, the number of pulls given to arm i in the play through history. Similarly, this algorithm has been applied in the literature to tree-based search algorithms, taking the name of Upper Confidence bound applied to Trees (UCT), introduced in [26]. Accordingly, random sampling is coupled to a tree-based search algorithm for a more efficient search in a defined space using an upper confidence bound algorithm. This leads to the selection of the most promising node of the tree, from which a sequence of actions is unrolled. The confidence bound for a parent node u and child node u_i of a tree search is given by

$$\text{UCT}(u_i, u) = \frac{Q(u_i)}{N(u_i)} + c\frac{\ln N(u)}{N(u_i)}, \quad (2)$$

where $Q(u_i)$ corresponds to the total simulation reward for the node u_i, and $N(\cdot)$ identifies the number of visits to a node. Although several versions of UCB have been proposed, as in [27], the algorithm's core takes advantage of the exploration-exploitation trade-off to find optimistic solutions for the choice of next moves in a task.

Upon methods selected from exploration-exploitation strategies and Monte Carlo sampling, we introduce a novel strategy for the same problem on Replay Buffering transitions sampling, to be specific a ***M***onte Carlo ***E***xploration-***E***xploitation ***T***rade-off (MEET) for an improved buffer sampling strategy. Instead of sampling on a state space, MEET samples transitions in the replay buffer with an exploration-exploitation strategy, which adapts well to the confidence of the network in solving the task. To express the mean and variance related to the exploration-exploitation strategy, we adopt the multi-head bootstrap network proposed in [4] for estimating uncertainty in the Q-Value estimation (i.e., $\hat{Q}$). In our case, the multi-head uncertainty corresponds to the variance in the prediction of the Q-Value, as a result of the training of multiple Q-Heads. The process is described in Section 2.2.

To trade-off exploration and exploitation of transitions, we elaborate a priority score p for sampling a transition, i.e.,

$$p = \sigma^2(\hat{Q})\left(\mu(\hat{Q}) + \frac{1-\mu(\hat{Q})}{N(v)}\right), \quad (3)$$

where the value of p is computed for each transition stored in the buffer.

The number of visits for a transition v is expressed by $N(v)$. We can rewrite the same formula as:

$$p = \left(1 - \frac{1}{N(v)}\right)\mu(\hat{Q})\sigma^2(\hat{Q}) + \frac{1}{N(v)}\sigma^2(\hat{Q}), \quad (4)$$

where the exploitation corresponds to the first term, while the exploration is expressed with the second term . We notice that similarly to UCB1 and UCT bounds, both the exploitation and exploration parts consider the number of visits $N(v)$.

Intuitively, if the transition has not been sampled sufficiently, the variance term will encourage exploration. The variance would be higher as the Q-Value estimation of the multiple heads disagree on the unseen sample.

The more visits to the transition, the more relevant becomes the exploitation term. In this case, the multiplication of the transition's Q-Value mean and its variance gives a higher sampling priority score. In fact, while the Q-Value mean identifies the value of the action in that state, the variance multiplication assesses the uncertainty on the Q-Value for the given scene. Transitions with higher uncertainty and expected Q-Value are favored by the exploitation term, thus encouraging visits on promising scenes while multiple heads have not reached a consensus on the Q-Value estimate.

3.2. MEET

Algorithm 1 shows the complete pseudocode on a critic network. During the training of the critic network, at each step of the episode, we store a transition with an associated sampling score equal to the maximum among the already stored ones, according to Equation 3. In this step, we ensure that new injected transitions can be explored in the sampling process. Afterward, as proposed in [4], we randomly select some critics' heads so that only a subset of them, with probability m_p, are trained during each epoch. These are called active heads and are represented as $\tilde{Q}$. Transitions are sampled during the replay period following the normalized priority score shown in Equation 3.

The mean $\mu(\hat{\tilde{Q}})$ and variance $\sigma^2(\hat{\tilde{Q}})$ of the active heads are computed and normalized for stability reasons. These are needed to calculate the exploration-exploitation term. To use a 'sum-tree' implementation, the mean is constrained to be a positive number, thus shifted by the smallest mean observed so far. The computational complexity of MEET using the 'sum-tree' is $\mathcal{O}(\log N)$, where N identifies the number of samples in the buffer.

After the computation of the critic loss, we update the active critic heads with the gradient. To reduce the more likely sampled transitions bias, the gradient is normalized with the factor $1/N(v)$. This factor represents the Monte-Carlo estimation of the transitions bias.

In practice, the MEET algorithm can be applied to any DRL method using critic networks and can be extended to discrete action space problems without loss of generality.

4. EXPERIMENTS

In this section, we first review the implementation settings. Afterwards, we benchmark the proposed sampling strategy against state-of-the art sampling strategies on MuJoCo, a continuous control benchmark suite.

4.1. Implementation Settings

In the implementation, we used PyTorch v1.8.0.™- GPU v2.4.0 with CUDA® Toolkit v11.1.0. As a processing unit, we used the Nvidia® Tesla® P40 GPU, Intel® Core i7-8700K

Algorithm 1: MEET Buffer Sampling

```
Steps $T$. Number of critic heads $L$. Initialize Replay
  Memory $\mathbf{H} = \emptyset, \triangle = 0, p_0 = 1, m_p$
Observe $s_0$ and choose $a_0 \sim \pi_\theta(s_0)$
for t=1 to T do
    Observe $s_t$, $r_t$
    Set the number visits: $N(v_t) = 0$
    Store transition $v_t = (s_{t-1}, a_{t-1}, r_t, s_t, N(v_t))$
    in $\mathbf{H}$ with maximal priority $p_t = \max_{i<t} p_i$
    Set Q-Head mask $m_t \sim B(1, m_p)$
    Number of heads $M = \sum_{l=1}^{L} m_t$
    Select active heads $\tilde{Q}$ based on $m_t$
    if $t \equiv 0 \; Mod \; K$ then
        for $j = 1$ to $k$ do
            Sample transition $j \sim P(j) = p_j / \sum_i p_i$
            Compute Q-Heads mean $\mu(\hat{\tilde{Q}})$
            Compute Q-Heads variance $\sigma^2(\hat{\tilde{Q}})$
            Update $N(v_j) = N(v_j) + 1$
            Update priority score:
            $p_j = \sigma^2(\hat{\tilde{Q}}) \left( \mu(\hat{\tilde{Q}}) + \frac{1 - \mu(\hat{\tilde{Q}})}{N(v_j)} \right)$
            for m in M do
                $\hat{y} = r + \gamma \cdot \hat{\tilde{Q}}_{m,target}(s_{j+1}, a_{j+1})$
                Compute Critic Loss: $J_{m,,j}$
            end
        end
        $J = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{k} \sum_{j=1}^{k} J_{m,j}$
        Accumulate weight-change
        $\triangle \leftarrow \triangle + \frac{1}{N(v_j)} \cdot \nabla_\theta J$
        Update weights $\theta \leftarrow \theta + \eta \cdot \triangle$
    Choose action $a_t \sim \pi_\theta(s_t)$
end
```

CPU, and DIMM 16GB DDR4-3000 module of RAM. The algorithms are evaluated over nine publicly available continuous control environments. The results are averaged over five experiments per environment and the code is published on Github [2].

MEET outperforms state-of-the-art DRL buffer sampling strategies for dense rewards. Figures 1 illustrate the average evaluation reward attained by the soft-actor critic on different MuJoCo tasks. The MEET sampling consistently surpasses the uniform and prioritized buffer in terms of convergence and on average peak performance by 26%. As pointed out in [28], the prioritized buffer can be detrimental in continuous control problems: This characteristic is also observed in our experiments. The advantage of MEET is well underlined in the environment with the largest action space, Humanoid-v3, by the increasing performance gap. In addition, the soft-actor critic is not saturating with MEET sampling.

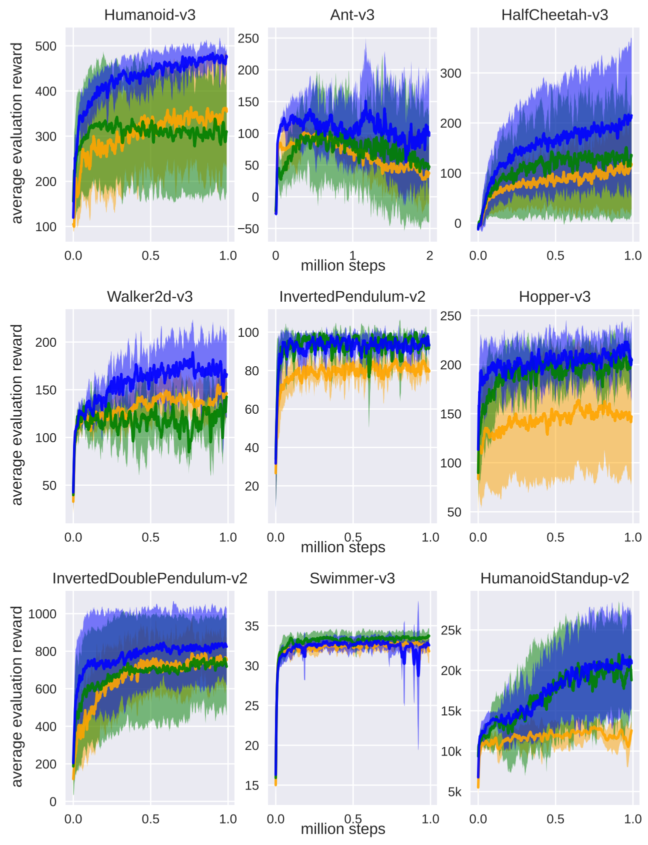

Fig. 1: Comparison of MEET (blue), Uniform (green), and Prioritized (orange) Sampling Strategies on MuJoCo

5. CONCLUSION

The main contribution of this paper is a method for prioritized sampling of transitions that trades off exploration-exploitation. To this end, the uncertainty estimation of the Q-Value function is used to sample more relevant transitions for the learning process. The experiments show that the presented algorithm outperforms existing methods on simulated scenarios. When benchmarked on the MuJoCo simulation environments, the MEET sampling consistently outperforms existing methods on convergence speed and performance by 26%. This paper evaluates MEET's performance on classical control tasks in which the action space is continuous. In future work, we expect to evaluate our approach in a discrete action space and with sparse rewards. Especially for the latter, we believe that the exploitation term is beneficial.

6. REFERENCES

[1] William Fedus et al., "Revisiting Fundamentals of Experience Replay," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3061–3071.

[2] Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause, "Information-

[2] `github.com/juliusott/uncertainty-buffer`

directed exploration for deep reinforcement learning," *International Conference on Learning Representations*, 2019.

[3] Andreas Sedlmeier et al., "Uncertainty-Based Out-of-Distribution Classification in Deep Reinforcement Learning," in *International Conference on Agents and Artificial Intelligence (ICAART)*, 2022, pp. 522 – 529.

[4] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy, ," in *Deep Exploration via Bootstrapped DQN*. 2016, p. 4033–4041, Advances on Neural Information Processing Systems (NIPS).

[5] David Silver et al., "Deterministic Policy Gradient Algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[6] Timothy P. Lillicrap et al., "Continuous Control with Deep Reinforcement Learning," in *4th International Conference on Learning Representations ICLR*, 2016.

[7] Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine, "Continuous Deep Q-Learning with Model-based Acceleration," in *International conference on machine learning*. PMLR, 2016, pp. 2829–2838.

[8] Tuomas Haarnoja et al., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[9] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[10] Volodymyr Mnih et al., "Human-level Control through Deep Reinforcement Learning," *Nat.*, vol. 518, no. 7540, pp. 529–533, 2015.

[11] Long Ji Lin, "Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching," *Machine Learning*, vol. 8, pp. 293–321, 1992.

[12] Tom Schaul et al., "Prioritized experience replay," *CoRR*, vol. abs/1511.05952, 2016.

[13] Marc Brittain et al., "Prioritized Sequence Experience Replay," *CoRR*, vol. abs/1905.12726, 2019.

[14] Jose A Arjona-Medina et al., "RUDDER: Return Decomposition for Delayed Rewards," in *Advances in Neural Information Processing Systems*, 2019, vol. 32.

[15] Isack Thomas Nicholaus and Dae-Ki Kang, "Robust Experience Replay Sampling for Multi-Agent Reinforcement Learning," *Pattern Recognition Letters*, vol. 155, pp. 135–142, 2022.

[16] Daochen Zha et al., "Experience Replay Optimization," in *International Joint Conference on Artificial Intelligence*, 2019, pp. 4243–4249.

[17] Gaon An et al., "Uncertainty-based offline reinforcement learning with diversified q-ensemble," vol. 34, 2021.

[18] Vincent Mai et al., "Sample Efficient Deep Reinforcement Learning via Uncertainty Estimation," *arXiv e-prints*, pp. arXiv–2201, 2022.

[19] Björn Lütjens et al., "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8662–8668.

[20] Gabriel Kalweit and Joschka Boedecker, "Uncertainty-driven imagination for continuous deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2017, pp. 195–206.

[21] Xue Bin Peng et al., "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[22] Yaniv Ovadia et al., "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift," in *Advances in Neural Information Processing Systems*. 2019, vol. 32, Curran Associates, Inc.

[23] Alex Kendall and Yarin Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?," in *Advances on Neural Information Processing Systems*. 2017, p. 5580–5590, Curran Associates Inc.

[24] Ian Osband, John Aslanides, and Albin Cassirer, "Randomized Prior Functions for Deep Reinforcement Learning," in *Advances on Neural Information Processing Systems*. 2018, p. 8626–8638, Curran Associates Inc.

[25] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[26] Levente Kocsis and Csaba Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[27] Peter Auer, "Using Confidence Bounds for Exploitation-Exploration Trade-Offs," *Journal of Machine Learning Research*, vol. 3, pp. 397–422, 2003.

[28] Gabriel Barth-Maron et al., "Distributed distributional deterministic policy gradients," *International Conference on Learning Representations*, 2018.