Value-based Reinforcement Learning using Efficient Hyperdimensional Computing

Yang Ni[1], Danny Abraham[1], Mariam Issa[1], Yeseong Kim[2], Pietro Mercati[3],and Mohsen Imani[1*]
[1]University of California Irvine, [2]Daegu Gyeongbuk Institute of Science and Technology, [3]Intel Labs

[*]Email: m.imani@uci.edu

***Abstract*—Reinforcement Learning (RL) has opened up new opportunities to solve a wide range of complex decision-making tasks. However, modern RL algorithms, e.g., Deep Q-Learning, are based on deep neural networks, resulting in high computational costs. In this paper, we propose QHD, an off-policy value-based Hyperdimensional RL, that mimics brain properties toward robust and real-time learning. QHD relies on a lightweight brain-inspired model to learn an optimal policy in an unknown environment. We first develop a novel mathematical foundation and encoding module that maps state-action space into high-dimensional space. We accordingly develop a hyperdimensional regression model to approximate the Q-value function. QHD-powered agent makes decisions by comparing Q-values of each possible action. QHD provides 34.6× speedup and significantly better quality of learning than deep RL algorithms.**

I. Introduction

Reinforcement Learning (RL) has opened up new opportunities to solve a wide range of complex decision-making tasks that were previously out of reach for a machine. Compared to supervised and unsupervised learning, RL does not have direct access to labeled training data. Instead, it trains a self-learning agent using the observations of states and feedback rewards from the environment. The learning process of RL is similar to how humans learn to perform new tasks.

RL methods are categorized into policy-based and value-based RL. The policy-based method directly parameterizes the policy. Value-based RL supports off-policy training, i.e., all past interactions can be used toward learning. Therefore, it is much more sample-efficient compared to the policy-based method. Q-learning is one of the most popular value-based reinforcement learning methods. Deep Q-Networks (DQN) exploits DNN to learn an approximation of Q-value for every pair of action and state. DQN learns complex tasks without modeling the environment, but its power comes at a price, i.e., the huge computation cost and long learning time. This makes it only suitable for powerful computers in the cloud rather than local devices with fewer computing capabilities.

Therefore, we redesign the RL algorithm with the brain-inspired HyperDimensional Computing (HDC) [1]. Compared to DNN, HDC is highly efficient and robust against noise. HDC is motivated by the fact that brains express information using a vast number of neurons. For inputs in the lower-dimensional space, HDC usually encodes them to vectors of several thousand dimensions, i.e., hypervectors. The learning process is based on highly-parallelizable element-wise operations of hypervectors. HDC has been applied as a lightweight machine learning solution to multiple applications, where it is capable of achieving comparable accuracy to DNN with significantly higher efficiency.

Current HDC solutions mainly focus on traditional classification and clustering. In contrast, QHD is a value-based Hyperdimensional RL algorithm with off-policy training. The main contributions of the paper are listed as follows:

- To the best of our knowledge, QHD is the first off-policy value-based hyperdimensional reinforcement learning algorithm targeting discrete action space. QHD relies on lightweight HDC models to learn an optimal policy in an unknown environment. Our neural-inspired QHD uses operations that are extremely hardware-friendly.
- We evaluate the effect of the different RL training batch sizes and local memory capacity on the QHD quality of learning. QHD is able to utilize even a small amount of available training data. Our QHD is also capable of online learning with a tiny local memory capacity. QHD provides real-time learning by further decreasing the memory capacity and batch size.

II. Related Work

Reinforcement Learning: In recent years, RL algorithms have obtained dramatically more attention because of the advancement in Deep RL. Algorithms like DQN greatly expand the application of Deep RL to fields like computer games [2], transportation optimization [3], and health care [4]. These works utilize DNN to handle complex agent-environment interactions. However, the frequent DNN model update during the RL training process is computationally intensive with insufficient efficiency.

Hyperdimensional Computing: Prior HDC works mainly provide solutions to classification and cognitive tasks, such as knowledge graph processing [5], genomic sequencing [6], and speech recognition [7]. In these highlighted machine learning applications, HDC has outperformed state-of-the-art machine learning solutions, e.g., support vector machines and neural networks. Recent orthogonal work proposes an HDC-based policy-based RL specifically for continuous control tasks [8]. However, this work does not provide support for RL tasks with discrete action space. In addition, policy-based RL methods are less sample-efficient due to the lack of off-policy training.

III. QHD: Hyperdimensional Q-learning

Fig. 1 shows an overview of QHD supporting hyperdimensional reinforcement learning. In Fig. 1(a), the Cartpole example illustrates two components (Agent and Environment) and three variables (Action, State, and Reward) in common RL tasks. The cartpole is the agent, and the space around it is the environment. For each step, the cart chooses an action, then the state of the cart and pole will be updated accordingly. The

978-3-9819263-7-8/DATE23/© 2023 EDAA

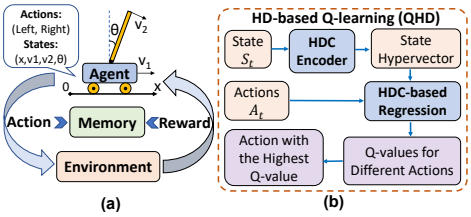

Fig. 1. Overview of QHD reinforcement learning.

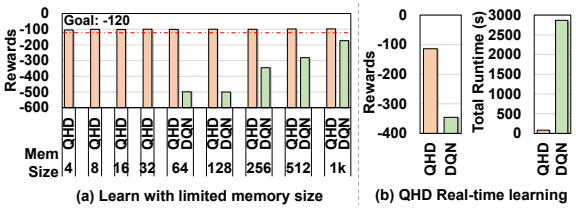

Fig. 2. QHD learning efficient with tiny local memory.

interaction between the agent and environment forms a loop in which the action taken based on the current state leads to the next state and reward. On the other hand, we have a trajectory composed of different states, rewards, and actions from each time step until the pole loses balance. The trajectory of each episode is saved in local memory for later learning.

Fig. 1(b) shows QHD algorithm guiding the agent in the decision-making process. In each time step, we map the state vector to a holographic hypervector using the HDC encoder. The hyperdimensional regression algorithm then predicts the Q-value for each possible action based on the input state hypervector. The final step of QHD chooses an action with the highest Q-value.

In QHD algorithm, the learning process starts by mapping the current state vector from the original to high-dimensional space, i.e., hypervector encoding. The input to our hyperdimensional regression model is the action for evaluation and the encoded state hypervector. The output is the action-state value or Q-value. Our regression consists of multiple model hypervectors; for the evaluation of each action, we only select one of the model hypervectors that corresponds to it. Even though RL is not a typical supervised learning task, the regression part of it trains under supervision. The true value is given by the ideal Q-function, and we use hyperdimensional regression to approximately calculate the Q-value. For model updates, we either add or subtract a portion of the state hypervector to the model, weighted by the regression error. The lightweight component-wise operations in our regression design contribute to the fast learning process for QHD.

We use a greedy policy that prefers actions with higher Q-values. However, it is crucial to balance the exploration of the environment and the exploitation of the learned model. Therefore, we combine a random exploration strategy with the greedy policy, i.e., the ϵ-decay policy. The probability of selecting random actions will gradually drop after the agent explores and learns for several episodes. Once an action is chosen by QHD, the agent interacts with the environment. We then obtain the new state for the agent and the feedback reward from the environment. This chain of actions and feedbacks form a trajectory or an episode until some termination conditions are met. To train an RL algorithm, these episodes or past experiences are usually saved to local memory as training samples. The objective of QHD is to achieve optimal policy and maximize the accumulated rewards within one episode. Since most RL tasks can be viewed as a Markov Decision Process (MDP), the Bellman equation gives a recursive expression for the Q-value at step t, the expected sum of current rewards, and the Q-value for step $t+1$. After obtaining the predicted Q-value and true Q-value from the regression model and the bellman equation, we perform regression model updates. We update the model corresponding to the action taken, using the regression error $q_{t\_true} - q_{t\_pred}$ and the encoded state hypervector: $\vec{\mathcal{M}}_{A_{t+1}} = \vec{\mathcal{M}}_{A_t} + \beta(q_{t\_true} - q_{t\_pred}) \times \vec{\mathcal{S}}_t$.

Usually, the local memory for RL experience replay is designed with infinite capacity, i.e., the agent has access to all previous experiences during the training. However, in practice, the memory capacity is limited due to energy and space budgets, especially in the low-power edge environment. Thus, in Fig. 2(a), we evaluate the performance of our QHD with limited memory size and compare it to the DQN results. The figure shows that DQN performs poorly when the memory size is 64 and 128, with an average reward of -500. However, our QHD can reach that goal even with a memory size as large as its batch size. These results show that QHD can perform RL tasks with online learning, i.e., a tiny local memory.

We also take one step further to explore the QHD capability of real-time learning. We set both the batch and memory sizes to 1, which means the agent will learn based on only the current sample, without any access to previous experiences. We use DQN with 256 memory size and 64 batch size as an online-learning comparison since DQN does not learn with a real-time setting. As shown in Fig. 2(b), even with larger memory size and batch size, DQN achieves significantly lower rewards (-345.4). For a 500-episode training, our QHD achieves average rewards of -113.7 using 83 seconds, which leads to a 34.6× speedup in total runtime.

Acknowledgements

This work was supported in part by National Science Foundation #2127780, Semiconductor Research Corporation (SRC) AI Hardware and Hardware Security, Office of Naval Research, grants #N00014-21-1-2225 and #N00014-22-1-2067, the Air Force Office of Scientific Research under award #FA9550-22-1-0253, and a generous gift from Cisco.

References

[1] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, no. 2, pp. 139–159, 2009.

[2] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] J. Ke *et al.*, "Optimizing online matching for ride-sourcing services with multi-agent deep reinforcement learning," *arXiv preprint arXiv:1902.06228*, 2019.

[4] H.-H. Tseng *et al.*, "Deep reinforcement learning for automated radiation adaptation in lung cancer," *Medical physics*, vol. 44, no. 12, pp. 6690–6705, 2017.

[5] P. Poduval *et al.*, "Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning," *Frontiers in Neuroscience*, p. 5, 2022.

[6] Z. Zou *et al.*, "Biohd: an efficient genome sequence search platform using hyperdimensional memorization," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 656–669.

[7] A. Hernandez-Cane *et al.*, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in *DATE*. IEEE, 2021, pp. 56–61.

[8] Y. Ni *et al.*, "Hdpg: hyperdimensional policy-based reinforcement learning for continuous control," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1141–1146.