DHN: Deep Hamiltonian Network for Variational Reinforcement Learning

Zeliang Zhang[1], Yipeng Wang[2], Zeqi Liu[2], Xiao-Yang Liu[2]

[1]Huazhong University of Science and Technology, Wuhan, China.
[2]Columbia University, New York, US.

Introduction

Deep variational reinforcement learning by optimizing Hamiltonian equation is a novel training method in reinforcement learning. Liu [1] proposed to maximize the Hamiltonian equation to obtain the policy network. In this poster, we apply the massively parallel simulation to sample trajectories (collecting information of the reward tensor) and train the deep policy network by maximizing a partial Hamiltonian equation. On the FrozenLake 8×8 and GridWorld 10×10 examples, we verify the theory in [1] by showing that deep Hamiltonian network (DHN) for variational reinforcement learning is more stable and efficient than DQN [2]. Our codes are available at [3].

Deep Hamiltonian Network

We denote the state, action and reward of the k-th step/transition in a trajectory as s_k, a_k and r_k, respectively. We denote the set of states and actions as $\mathbb{S}$ and $\mathbb{A}$. $\boldsymbol{\pi}(s, a)$ denotes the probability of taking action a at state s. We denote the partial inner product $< \boldsymbol{x}, \boldsymbol{y} >_\Omega = \sum_{i \in \Omega} \boldsymbol{x}_i \boldsymbol{y}_i$, where Ω is an index set.

Reward Tensor: For all the possible k-step trajectories, we use a reward tensor $\mathcal{C}^{(k)} \in \mathbb{R}^{|\mathbb{S}\times\mathbb{A}|^k}$ to record the total rewards associated with k-step transitions.

Hamiltonian Equation:

$$H = \sum_{k=1}^{K\to\infty} \langle \mathcal{C}^{(k)}, \underbrace{\pi \otimes \pi \otimes \cdots \otimes \pi}_{k \text{ times}} \rangle, \tag{1}$$

which is the inner product of a reward tensor $\mathcal{C}^{(k)}$ and the k-times outer product of π.

Basic Idea: It is difficult to obtain the full accurate reward $\mathcal{C}^{(k)}$, $k = 1, 2, 3..., K \to \infty$, for problems with medium size state and action spaces, because the size of $\mathcal{C}^{(k)}$ grows exponentially with k. Instead of obtain $\mathcal{C}^{(k)}$, we utilize massively parallel simulations to sample $\mathcal{C}^{(k)}$, $k = 1, 2, ..., K$, and directly train a deep policy network to learn the policy π with the target function as the partial of (1), i.e.,

$$H_\Omega = \sum_{k=1}^{K} \langle \mathcal{C}^{(k)}, \underbrace{\pi \otimes \pi \otimes \cdots \otimes \pi}_{k \text{ times}} \rangle_{\Omega_k}, \tag{2}$$

where Ω_k is the index set in $\mathcal{C}^{(k)}$, $k = 1, 2, ..., K$.

Training Process: As shown in Fig. 1, our training process for the deep Hamiltonian network (DHN) consists of the following three steps,

- 1) Perform massively parallel simulations and obtain random trajectories.
- 2) Fetch random transition batches and feed them to the deep policy network to obtain the corresponding probabilities.
- 3) Maximize (2) and utilize back propagation method to update the parameters of the deep policy network.

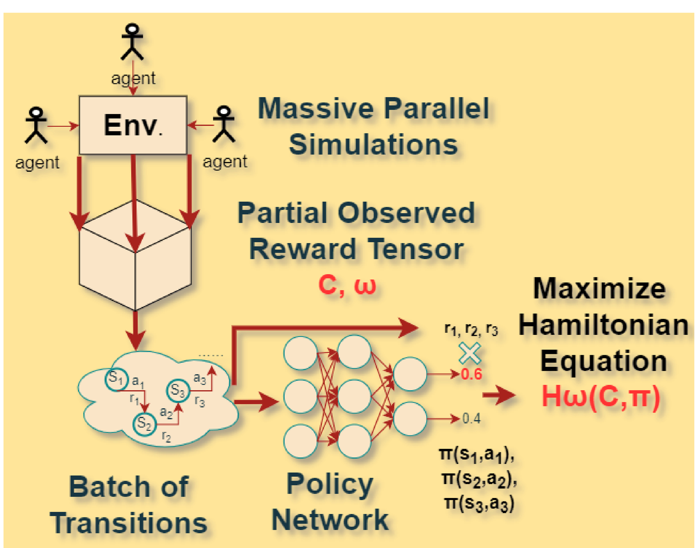

Fig. 1. The training process of DHN.

Frozen Lake

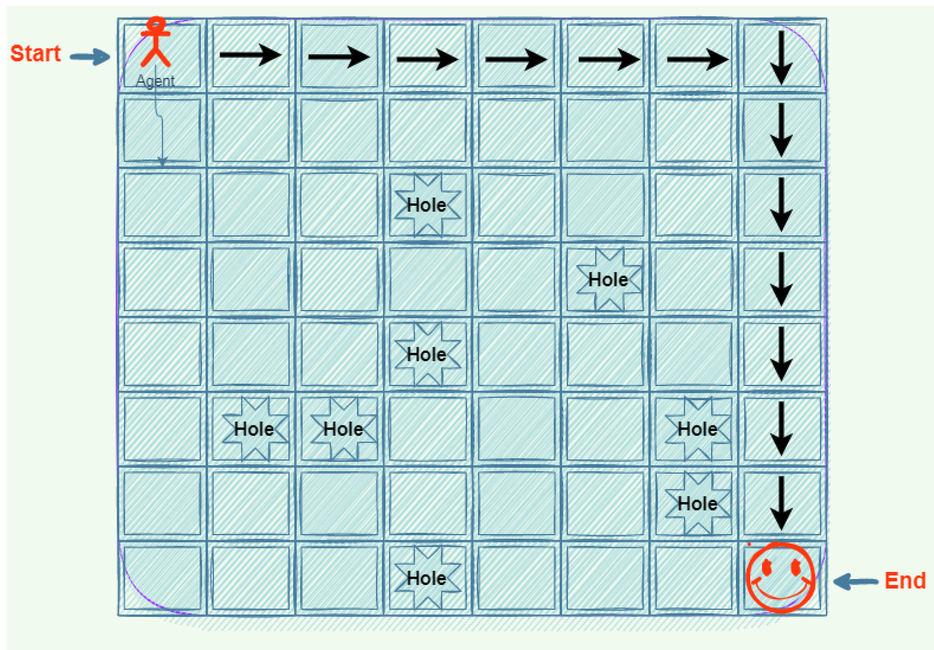

Fig. 2. The Frozen Lake 8×8 game.

Environment: Frozen Lake 8×8, a game in OpenAI Gym.
Rules: As shown in Fig. 2, the Frozen Lake has 8×8 states with 4 optional actions to move around. The agent needs to go from the start point and find the way to the destination in limited steps. There are 8 holes which can cause the agent to fail the game.
Experiment Settings: We take Deep Q-learning (DQN) [2] as our baseline of which the implementation is provided by ElegantRL library [4]. We use a 4-layer fully connected neural network as the deep policy network both in DQN and DHN. We use Adam optimizer with the learning rate as 1×10^{-3} and set the batch size as 100.
Evaluation: We evaluate the performance of policy by computing the success rate, in which we use 50 agents to walk 100 steps and compute the rates of agents who successfully arrive the destination.
Result: Fig. 3 shows the success rate of agents with increasing the number of transitions learned by the network. compared with DQN, DHN has a more stable training process. It is easy for DQN to quickly obtain a good policy to win the game. But with increasing the number of transitions fed to the network, the performance of DQN shows a large and frequent shock while the performance of DHN shows the strong stability.

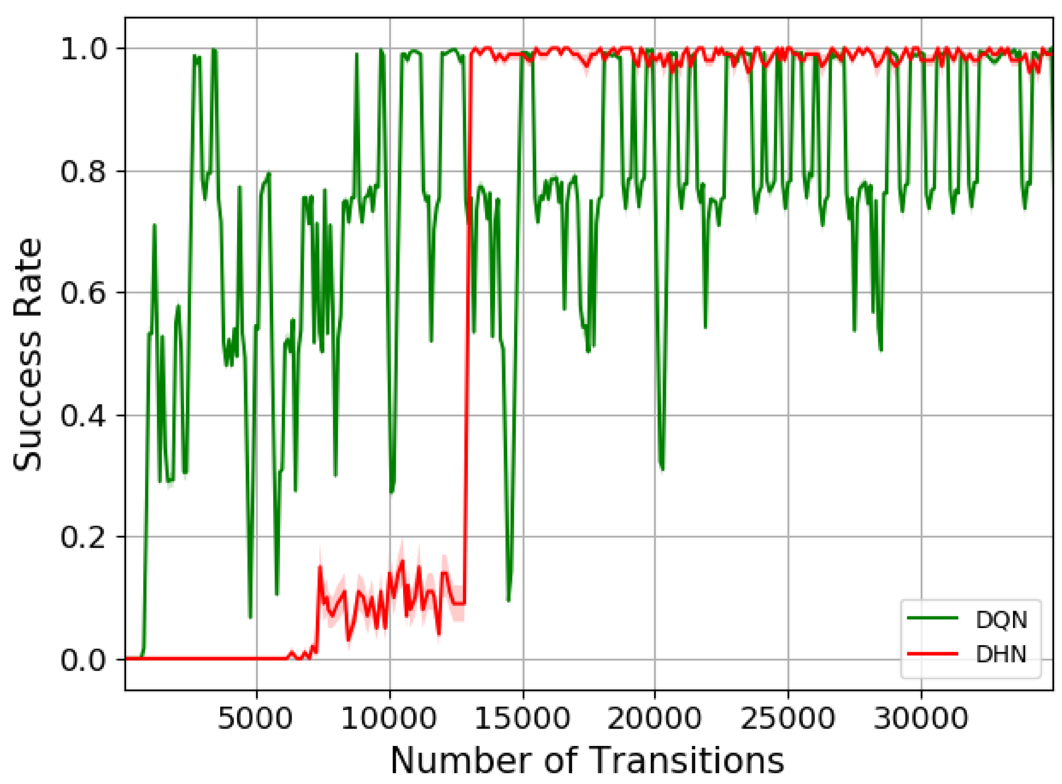

Fig. 3. Comparison of the results on Frozen Lake.

Summary of Environments

Table 1. States and actions in our experiments.

Tasks	State Vector	Action Vector
Frozen Lake	8×8	4
Grid World	10×10	4

Grid World

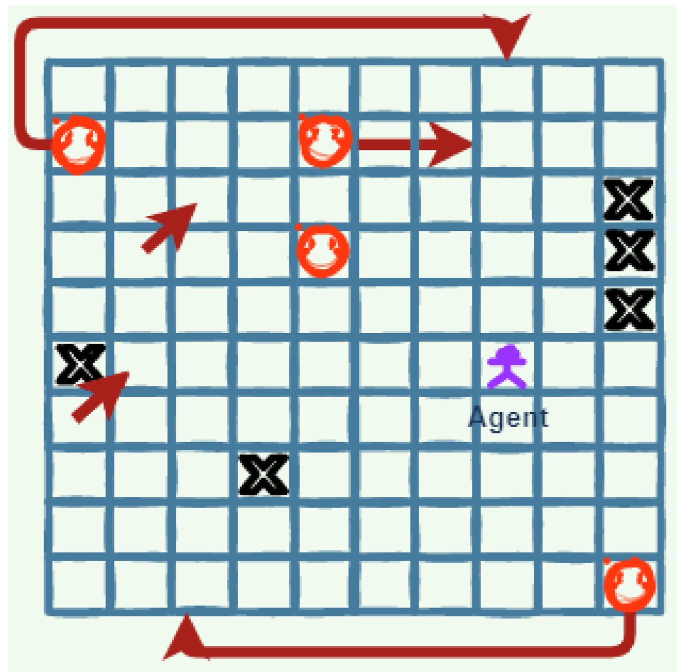

Fig. 4. The Grid World 10×10 game.

Environment: Grid World 10×10, a game available in our code.
Rules: As shown in the Fig. 4, the Grid World has 10×10 states with 4 optional actions to move around. The agent will initialize at a random locations and it needs to find the smiley as many as possible which has 10 reward in turn. It should be noted that there are some endpoints which may cause the agent game over and some transfer-points which transfer the agent to certain location.
Experiment Settings and Evaluation: Both the experiment settings and evaluation method are the same with that on Frozen Lake 8×8 game.
Result: Fig. 5 shows the mean reward obtained by the agents with increasing the training time. Compared with DQN, DHN has a faster training process. It only needs massive random parallel samples of trajectories and do not need any policy for guided sampling while DQN needs guided exploration in the training process which costs a large time consumption.

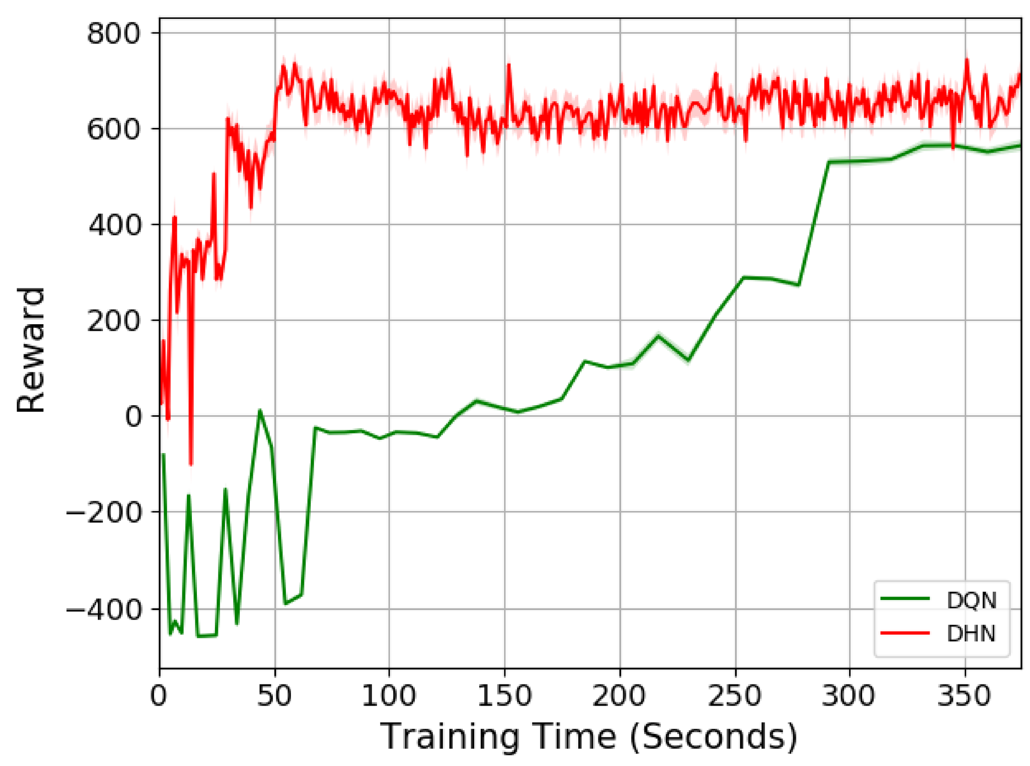

Fig. 5. Comparison of the results on Grid World.

Conclusions

In this poster, we propose to utilize massively parallel simulation to sample the reward tensor, and utilize deep policy network to learn the policy, thus estimate the Hamiltonian equation. We perform experiments, respectively on Frozen Lake 8×8 and Grid World 10×10, to further verify the theory of deep variational reinforcement learning by optimizing Hamiltonian equation. The results show that compared with conventional DQN method, the DHN is more stable and efficient.

References

[1] X.-Y. Liu and Y. Fang. Quantum tensor networks for variational reinforcement learning. *NeurIPS 2020 Workshop on Quantum Tensor Networks in Machine Learning.*
[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop, 2013.*
[3] https://github.com/AI4Finance-Foundation/Quantum-Tensor-Networks-for-Variational-Reinforcement-Learning-NeurIPS-2020.
[4] X.-Y. Liu, Z. Li, Z. Wang, and J. Zheng. ElegantRL: A scalable and elastic deep reinforcement learning library. https://github.com/AI4Finance-Foundation/ElegantRL, 2021.

Contact information:
Zeliang Zhang
v-zezhang@microsoft.com