An Improved Deep Reinforcement Learning with Sparse Rewards

Maxwell Hwang[1], Kao-Shing Hwang[2], Lu-Cheng Chi[2]
[1]School of Medicine, Zhejiang University,
Hangzhou, China.
himax26@126.com [2]Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
[1]hwang@ccu.edu.tw

***Abstract*- This paper presents an improved deep reinforcement learning which encourages an agent to explore unvisited states in an environment with sparse rewards. The improved method is based on an actor-critic approach. It uses some neglected observations from the background as the target output of supervised learning, providing the agent denser training signals to bootstrap reinforcement learning. Moreover, the improved method uses the prediction loss from supervised learning as feedback for the agent's exploration in the environment, called the label reward, to encourage the agent to explore unvisited states. Finally, the improved method constructs multiple neural networks to learn a policy by the Asynchronous Advantage Actor-Critic algorithm.**

Keywords-Reinforcement Learning, Actor-Critic algorithm, Asynchronous Advantage Actor-Critic algorithm, Supervised Learning, Sparse Rewards.

I. INTRODUCTION

In reinforcement learning [1], how an agent explores an environment with sparse rewards is a long-standing problem. Therefore, most of the reinforcement learning leads the agent to explore by designing denser rewards. However, poor reward design often results in a policy learned by the agent that is not the best solution.

Besides, in deep reinforcement learning [2], an agent relies on an image from an environment as an input to the neural network. However, some neglected observations from the environment, such as depth, might provide valuable information.

An improved deep reinforcement learning described in this paper, based on an actor-critic approach, uses the convolutional neural network as a heterogeneous encoder between the image and the neglected observation.

The improved method uses the neglected observation as the target output of supervised learning, providing the agent denser training signals to bootstrap reinforcement learning.

Hanjaya Mandala and Jacky Baltes are with the with the Department Electrical Engineering, National Taiwan Normal University, Taipei 10610, Taiwan (e-mail: ahanjaya@gmail.com and jacky.baltes@ntnu.edu.tw).

Saeed Saeedvand is with the Faculty of Electrical Engineering and Computer Engineering, University of Tabriz, Tabriz, 5166616471, Iran (e-mail: saeedvandr@tabrizu.ac.ir).

Moreover, the improved method uses the prediction loss from supervised learning as feedback for the agent's exploration in the environment, called the label reward to encourage the agent to explore unvisited states.

Finally, the improved method constructs multiple neural networks to learn a policy by the Asynchronous Advantage Actor-Critic algorithm.

II. BACKGROUND

A. Reinforcement Learning

In reinforcement learning, an agent interacts with an environment over several discrete time steps. At each time step t, the agent observes a state s_t and selects an action a_t. An agent is guided by a policy π, mapping from states s_t to actions a_t. After each action, the agent observes the next state s_{t+1} and receives feedback in the form of a reward r_t. This process continues until the agent reaches a terminal state or time limit.

The goal of learning is to find a policy π that maximizes the expected reward. In policy-based model-free methods, a function approximator, such as the neural network, computes the policy $\pi(a_t|s_t;\theta)$, where θ are parameters of the neural network. There are many methods for updating θ based on rewards received from the environment.

The REINFORCE method [3] uses gradient ascent on $\mathbb{E}(R_t)$.

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \quad (1)$$

where is the accumulated reward starting from time step t and increasingly discounted at each subsequent step by factor $\gamma \in (0, 1]$.

B. Actor-Critic Algorithm

The REINFORCE method updates θ using the gradient:

$$\nabla_\theta \log\pi(a_t|s_t;\theta)R_t \quad (2)$$

which is an unbiased estimator of $\nabla_\theta \mathbb{E}[R_t]$. The variance of the estimator is reduced by subtracting a baseline b_t and using the gradient:

$$\nabla_\theta \log\pi(a_t|s_t;\theta)\big(R_t - b_t(s_t)\big) \quad (3)$$

One common baseline is the value function defined as:

$$V_\pi(s_t) = \mathbb{E}[R_t|s_t] \quad (4)$$

which is the expected return for following the policy π in state s_t. In this approach, the policy and the baseline can be viewed as actors and critics in the Actor-Critic algorithm [3].

C. Asynchronous Advantage Actor-Critic Algorithm(A3C)

The Actor-Critic algorithm consists of two convolutional layers where the first layer has 16×8×8 filters and a stride of 4; the second one has 32×4×4 filters with a stride of 2. The CNN is followed by a fully-connected layer with 256 units; a rectifier nonlinearity follows each hidden layer. The two outputs are a softmax layer, which approximates the policy $\pi(a_t|s_t;\theta)$, and a linear layer to output an estimate of $V(s_t;\theta)$.

Two loss functions are associated with the Actor-Critic algorithm. For the policy, this is:

$$f_\pi(\theta) = \log\pi(a_t|s_t;\theta)\big(R_t - V(s_t;\theta_t)\big) \quad (5)$$

$$R_t = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_t) \quad (6)$$

where R_t is the estimated discounted reward in the time interval from t to $t+k$, and k is upper-bounded by t_{max}.

The loss function for the estimated value function is:

$$f_v(\theta) = \big(R_t - V(s_t;\theta)\big)^2 \quad (7)$$

A3C [4], based on an actor-critic approach, constructs multiple neural networks as multiple agents to learn a policy. Multiple agents play concurrently and update the weights of the neural network through asynchronous gradient descent. In an asynchronous training environment, an agent represents on its environment, sees observation and reward pairs, and takes actions that are different from the other agents. Meanwhile, the neural network weights are stored in a central parameter server. Agents calculate gradients and send updates to the server after every t_{max} actions, or when a final state is reached. After each update, the central server propagates new weights to agents to guarantee they share a joint policy.

Training is performed by collecting the gradients $\nabla\theta$ from both of the loss functions and using the RMSProp algorithm [5] as an optimization:

$$g' = \alpha g^2 + (1-\alpha)\nabla\theta^2 \quad (8)$$

$$\theta \leftarrow \theta - \eta\nabla\theta/\sqrt{g'+\epsilon} \quad (9)$$

III. AN IMPROVED DEEP REINFORCEMENT LEARNING

A. Nav A3C Algorithm

An improved deep reinforcement learning described in this paper is based on the Nav A3C algorithm [6], which is a variant of the A3C algorithm.

The Nav A3C algorithm (see Figure 1) employs a two-layer stacked LSTM after the convolution neural network. It expands observations of agents to include the action and the reward from the previous time step. The variant of the A3C algorithm opts to feed the previously selected action to the second recurrent layer. The first layer only receives the reward and postulates that the first layer might be able to make associations between reward and visual observations from an environment provided as context to the layer where the policy is computed.

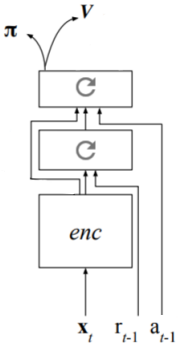


Figure 1: Nav A3C algorithm [6]

B. Deep Reinforcement Learning with Supervised Learning

An improved deep reinforcement learning described in this paper (see Figure 2), based on the Nav A3C algorithm, uses the convolutional neural network as a heterogeneous encoder between an image observation and other observations, which both are from an environment.

In an environment with sparse rewards, the improved method considers supervised learning from the convolutional neural network, using an environmental observation as the target output of supervised learning to compute the prediction loss, which provides an agent denser training signals to bootstrap reinforcement learning. The prediction loss is calculated on the current frame via a single layer MLP. Through denser training signals, the improved method accelerates the construction of the convolutional neural network in the environment with sparse rewards.

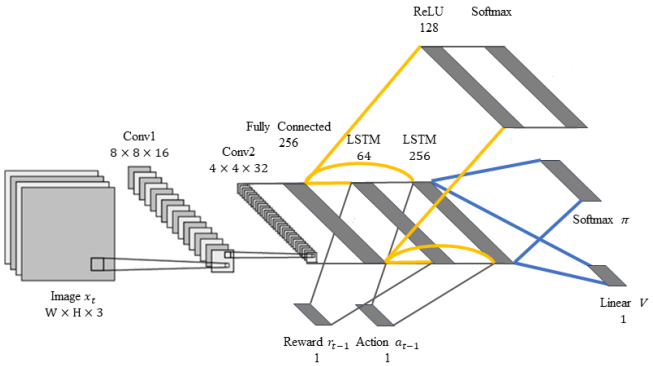


Figure 2: Neural network of an improved deep reinforcement learning.

C. Label Reward

An improved deep reinforcement learning described in this paper uses the prediction loss from supervised learning as feedback for an agent's exploration in an environment, called the label reward r_{label}, to encourage the agent to explore unvisited states.

In supervised learning, when the neural network is trained with the same training set, the prediction loss is gradually reduced by gradient descent. Therefore, an improved deep reinforcement learning described in this paper examines an agent's exploration in an environment by the prediction loss from supervised learning. During the process of learning, when the prediction loss is small, it indicates that the agent has visited the state

multiple times. When the predicted loss is substantial, the agent has not yet visited this state.

The improved method uses the label reward r_{label} with the reward $r_{environment}$, which is directly from the environment, as the new reward for updating weights of the neural network (see Figure 3).

$$r_t = r_{environment} + \beta r_{label} \quad (10)$$

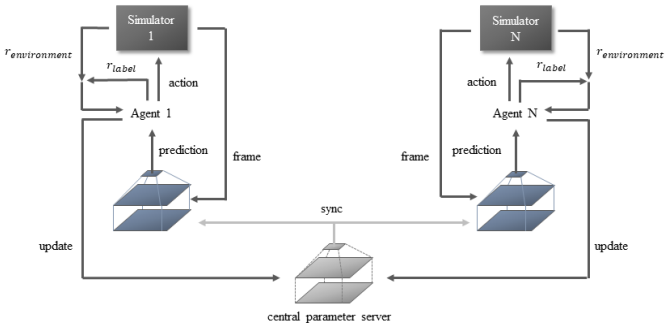


Figure 3: Architecture of an improved deep reinforcement learning.

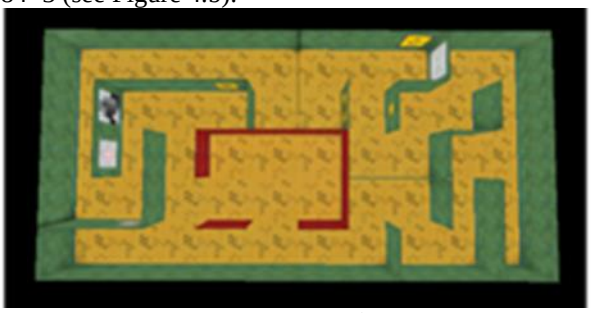
D. Training

An improved deep reinforcement learning described in this paper constructs eight neural networks as eight agents to learn a policy jointly by the Asynchronous Advantage Actor-Critic algorithm**.** Agents calculate gradients and send updates to the server after every 50 actions, or when a final state is reached.

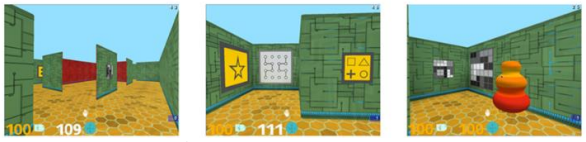
IV. EXPERIMENTS

A. First-person 3D maze

This paper considers a first-person 3D maze (see Figure 4.a) from the DeepMind Lab [7], which provides an agent with sparse rewards. The action space is discrete, comprising six actions: the agent can rotate in small increments, accelerate forward or backward, or sideways. The reward is achieved in this environment by reaching a goal from a random start location and orientation. If the goal is reached, the agent is respawned for a new start location and must return to the goal. The episode terminates when a fixed amount of time expires, providing the agent enough time to find the goal several times. There are sparse rewards. Apples are worth 1 point, and goals are 10 points. In the maze, the goal and fruit locations are fixed, and only the agent's start location changes. The image is 84×84×3 (see Figure 4.b).



(a)A Top-down view of a maze



(b) Images from the agent's egocentric viewpoint

Figure 4: A first-person 3D maze from the DeepMind Lab [7].

The DeepMind Lab simulates an agent finding a goal with an RGB-D camera in a first-person 3D maze. An improved deep reinforcement learning described in this paper uses depth information provided by the environment as the target output of supervised learning (see Figure 5).

The improved method uses a low-resolution variant of the depth map by reducing the screen resolution to 4×16 pixels. For the classification loss, the depth at each position is discretized into eight different bands. Figure 6 shows the learning curves.

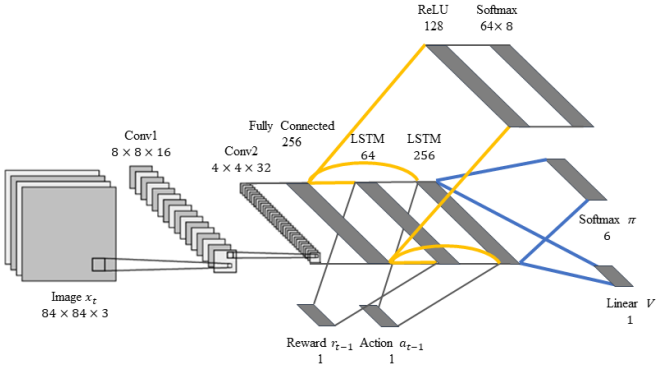
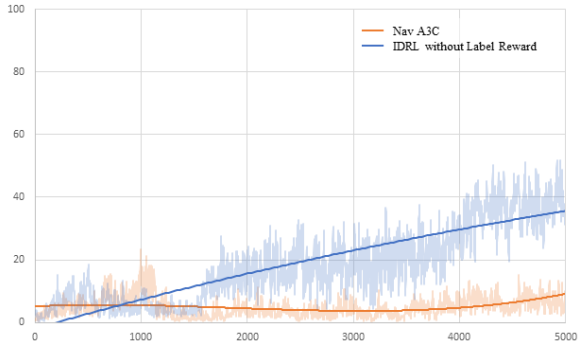
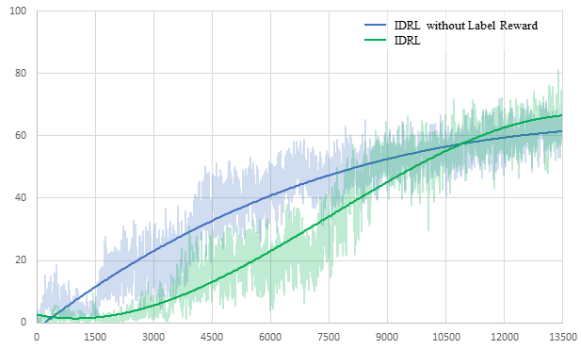


Figure 5: Neural network of an improved deep reinforcement learning for the maze.



(a)



(b)

Figure 6: Learning curves: (a) An improved reinforcement learning without the label reward, is compared with the Nav A3C algorithm. (b) An improved deep reinforcement learning is compared with the improved method without the label reward.

B. Wafer detection simulator

This paper also considers a simple 12×12 wafer detection simulator (see Figure 7) from the IRIS Lab, which provides an agent with sparse rewards. It attempts to find the shortest path of wafer detection. The action space is discrete, comprising 24 actions: the agent can move in 8 directions, and the maximum amount of movement for each direction is 3. Undetected chips in the wafer are represented by water blue. The upper limit of detection for each chip is 15. The number of detection for each chip is represented by a grayscale divided into 15 levels. The reward is achieved in this environment by the reward function:

$$\sum_{d=1}^{15} c_d \times 100 e^{-0.7d} \qquad (11)$$

After each action, the amount of detection for chips detected by the current action is *d*, and the number of those chips is c_d . The episode terminates when a fixed number of actions are taken, providing the agent enough time to detect the wafer several times. The image is 36×36×3 (see Figure 7).

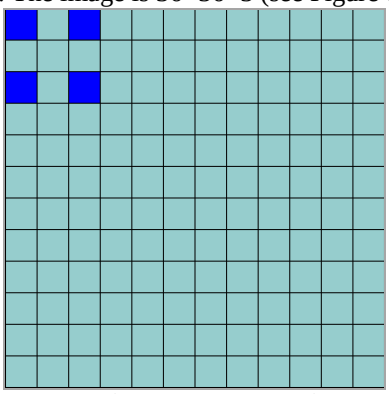


Figure 7: A simple 12×12 wafer detection simulator from the IRIS Lab.

The 12×12 wafer detection simulator from the IRIS Lab provides the amount of detection for each chip. An improved deep reinforcement learning described in this paper (see Figure 8) uses the amount of detection for each chip as the target output of supervised learning. The improved method reduces the amount of information to 8 values and considers a classification loss, where the amount of detection for each chip is discretized into ten different bands. Figure 9 shows the learning curves.

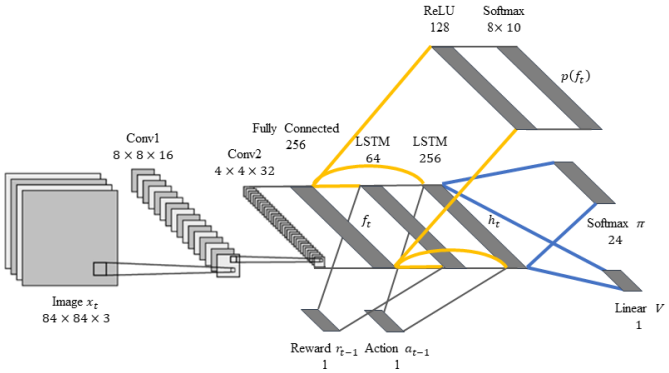


Figure 8: Neural network of an improved deep reinforcement learning for the wafer detection simulator.

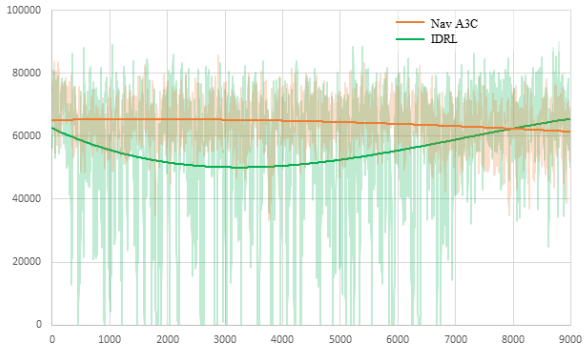


Figure 9: Learning curves: An improved reinforcement learning is compared with the Nav A3C algorithm.

V.CONCLUSION

The improved deep reinforcement learning described in this paper encourages an agent to explore in an environment with sparse rewards.

In section 4, this paper considers a first-person 3D maze from the DeepMind Lab, which provides an agent with sparse rewards, as a training environment for the agent. First, the improved method without the label reward is compared with the Nav A3C algorithm (see Figure 6.a). In the absence of the label reward, the improved method uses supervised learning to improve the agent's performance in the early stage significantly. Second, an improved deep reinforcement learning is compared with the improved method without the label reward (see Figure 6.b). After adding the label reward, it is evident that the agent tends to explore the environment in the early stage. However, the agent finally learns a better policy.

Besides, even in an environment with denser rewards, the improved deep reinforcement learning described in this paper still encourages an agent to explore. This paper also considers a simple 12×12 wafer detection simulator from the IRIS Lab, which provides an agent with denser rewards, as a training environment for the agent. An improved reinforcement learning is compared with the Nav A3C algorithm (see Figure 9). Because the reward function described in this paper is not well-designed, it results in a policy learned by the agent that is not the best solution. However, the agent can still explore through the improved method, and finally, take a better policy.

VI. REFERENCES

[1]. R. Sutton and A. Barto, *Introduction to Reinforcement Learning*, MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[2]. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level Control through Deep Reinforcement Learning", *Nature*, 518(7540):529–533, 2015.

[3]. R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning", *Machine Learning*, 8(3-4):229–256, 1992.

[4]. V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu,

"Asynchronous Methods for Deep Reinforcement Learning", *ArXiv preprint arXiv:1602.01783*, 2016.

[5]. T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[6]. P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al, "Learning to Navigate in Complex Environments", *ICLR*, 2017.

Beattie, Charles, et al. "DeepMind Lab" *ArXiv preprint*