Grid World Navigation

Haosen Xing, Xueqian Li
Carnegie Mellon University
Pittsburgh, PA 15213
haosenx@andrew.cmu.edu, xueqianl@andrew.cmu.edu

1 Introduction

There is an infinite grid world with a single agent roaming around. The world contains jellybeans, tongs, onions, diamonds and walls, and the agent could move forward, turn 90 degree left or turn 90 degree right. It will receive +20 reward immediately when eating a jellybean and only receive +100 for picking up a diamond if it has an empty tong. Our work is to use reinforcement learning system to solve the problem of agent navigation and largely leverage the reward it fetched.

Usually, the grid world is a natural environment for applying reinforcement learning algorithms to find an optimal path and policy for the agent to reach the desired goal grid in the least number of moves. Solving grid world navigation problem is geometrically intuitive and has high relevance to many real-world applications. In the delivering and transportation system, we can leverage the grid-world navigation to do the collaboration of the multi-agents using the synchronous A2C algorithm, plan for the optimal path for all possible agents. Also, this is helpful when dealing with the allocation of the materials and resources, which can also be applied to the service robot. We want to stress this problem as a good illustration of the robotic perceptions.

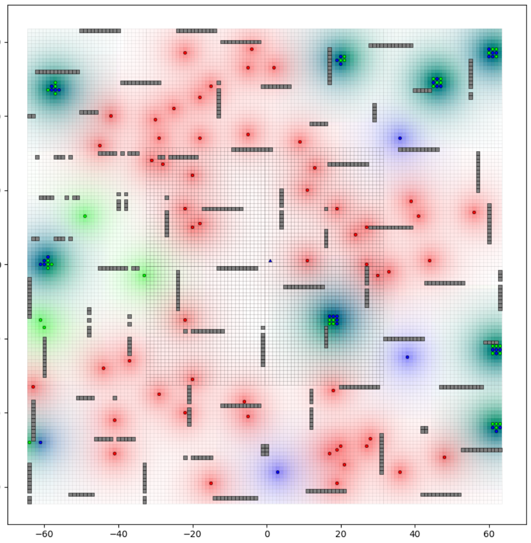


Figure 1: **Grid World Navigation Simulation Map** The red beaming square indicates Tong, and the blue one shows jellybeans, where the big darker green beaming pixels denotes the diamond. Besides, the gray strip here is the wall, and the small lighter green ones are what we don't care about.

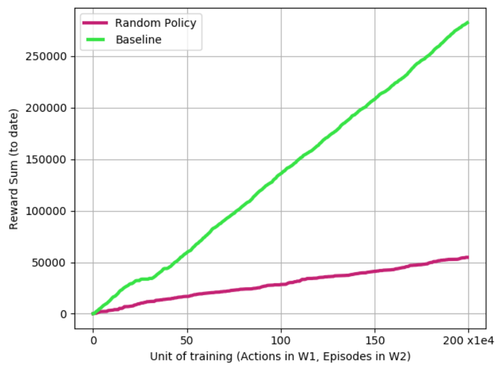
Preprint. Work in progress.

2 Background

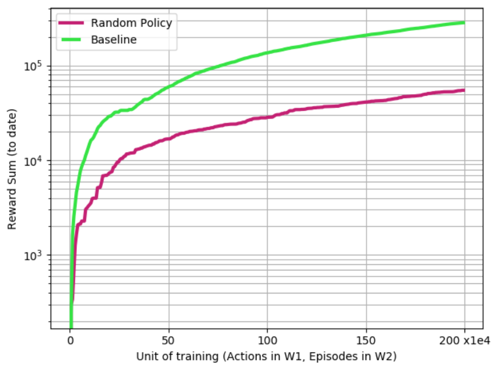
We used the Double Deep Q learning (Double DQN)[5] algorithm to implement the baseline work. Our Double DQN network has 4 fully connected layers, which we applied as $(3, 32)$, $(32, 64)$, $(64, 32)$, $(32, 3)$. It simply takes the current scent as the network input and outputs corresponding q-values in 3 possible directions. The target network is updated every 1000 steps and the action is taken in an *epsilon*-greedy fashion, for which the ϵ is decaying exponentially.

As shown in Fig. 2, in 2×10^6 steps, the total reward received by random policy is about 5×10^4 while in Double DQN, the reward reaches to around 2.7×10^5 . The reward obtained from our network is much better than the random choice policy. Meanwhile, from Fig. 2(cd), the variance of baseline network is much less.

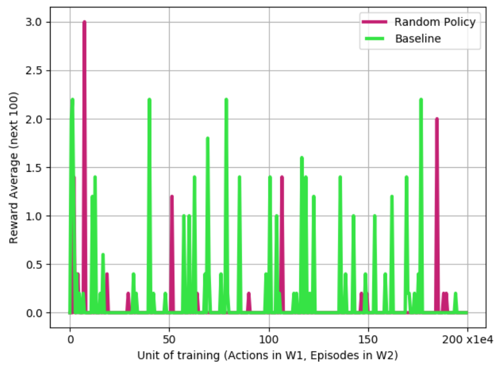
Thinking from another way, our baseline work just used the Double DQN, with input of the current scent, and we can get such a good results. It seemed that the grid-world is simple enough to use small network to train. After some tryouts about using the previous perceptions, we actually found the network is doing bad regarding this, so we threw away the idea of adding some previous perceptions. There are many other techniques we can utilize regarding this.



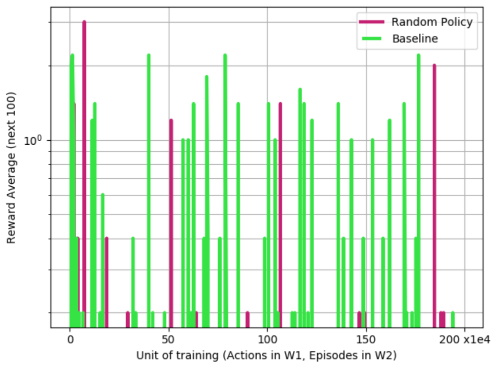
(a) linear cumulative



(b) log cumulative



(c) linear average



(d) log average

Figure 2: **Canonical plots for baseline implementation**

3 Related Work

Deep reinforcement learning allows the training of highly flexible and versatile deep neural networks to achieve action-selection policies for complex problems. [1] proposes a method which splits complicated navigation task into a number of manageable navigation behaviors and contains a domain randomization technique to guide the policy training in order to solve a complex mobile robot navigation problem. Gnanasekaran[2] applied compare Q-learning, approximate Q-learning and Deep Q-learning on the classical game Pacman based on total rewards and win-rate. In [3], a novel architecture named Imagination-Augmented Agents (I2As) combines model-free and model-based aspects in deep reinforcement learning. When there are multiple agents in the world requiring collaboration, He[4] presents a new model DRON (Deep Reinforcement Opponent Network), which has a policy learning module to predict Q-values and an opponent learning module to infer opponent strategy.

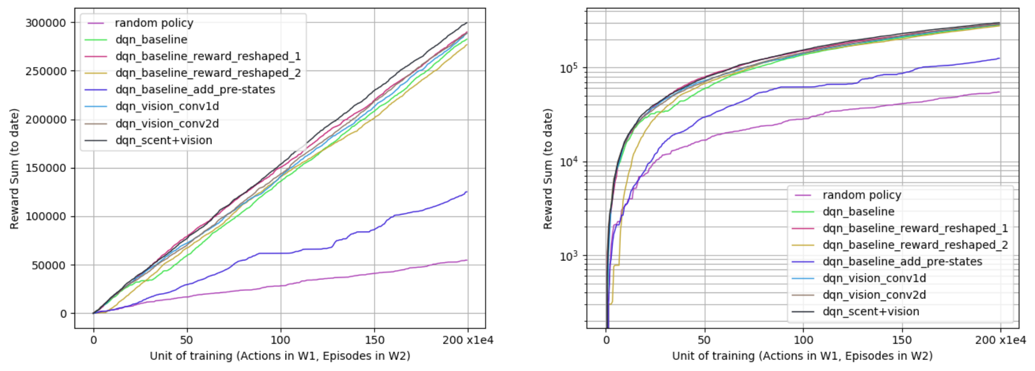
4 Methods

Double DQN Staying with Double DQN, we first tried to modify the observation input. (Since the baseline is pretty well, we thought that this is a good candidate method) In baseline, the input is only the current scent perception. Though we have tried 1 previous scent perception, we are not sure whether more previous knowledge would help us here. Therefore, we input a longer history of scent perception into the network. We concatenated 6 consecutively scent perceptions as a 1×18 vector into 4 fully connected layers. (The reason we chose 6 previous observations was that in this environment, the length of the regional grid-world the agent can perceive from vision sensor is 6 when counting himself/herself in.) The result surprisingly gets worse, which is shown as the purple line in Fig.3. The total reward in 2×10^6 steps is about 1.25×10^5 .

We also reshaped the reward function on the basis of the baseline to learn sub goals. Hitting the obstacle is bad, so we returned a reward of -10. Then we came up with two approaches of making reward reshaping. First reshaping is to make reward of picking up tongs equivalent to getting jellybeans, both yield to $+50$. In this way, if there are full of jellybeans, the agent may get more jellybeans to obtain higher reward. Our second thought is to emphasize more on picking up tongs to allow the network to learn to pursue more diamonds. The former has reward of $+100$, and the latter is $+20$. The result is shown in the Fig.3. We can clearly seen that after the reward reshaping settings like what we stated above, the agent got highest reward for the first reshaping, and actually performed worst for the second reshaping strategy. We thought that the reason behind is that we might want a "balance" here since there must be many jellybeans around, but with little diamonds.

After playing with the scent perception, we put the vision perception into both convolution $1D$ and $2D$ networks respectively. For the vision perception, the network is actually more difficult to train with the convolution layers steps in. The $1D$ neural network has 3 Conv1d layers and 4 fully connected layers. The $2D$ network has the same structure here. The results indicated that there are no big differences between them. We realized that this may due to the independent property of the scent perception for each specific pixels.

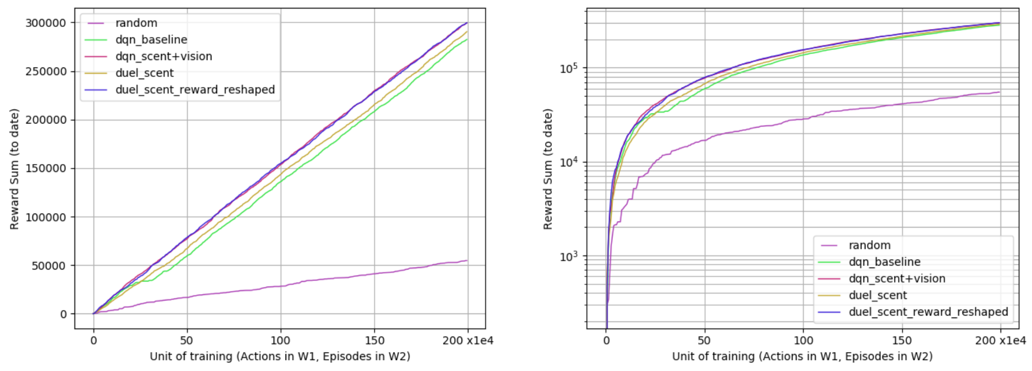
Finally, we concatenated the scent and vision into the network, which returns the highest reward. This also proved that more perceptions may produce better results.



(a) linear cumulative (b) log cumulative

Figure 3: **Cumulative Plots of trails in Double DQN**

Dueling Network Dueling Network[6] might be a better policy learning method in our scenario. The network learns action-independent value function and action-dependent advantage function. Unlike the Double DQN, Dueling Network will learn both the value and the related advantage. In this case, our network can improve the stability. For this experiment, we only used scent perception as the input, with 5 fully connected layers for both of them. The output dimension of the value network is 1, while of the advantage network is 3. And the result presented to be similar to the concatenation of scent and the vision perceptions, shown in Fig.4.



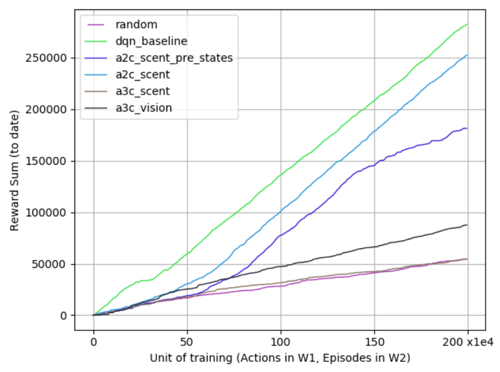
(a) linear cumulative (b) log cumulative

Figure 4: **Cumulative Plots of Dueling Networks with Double DQN**

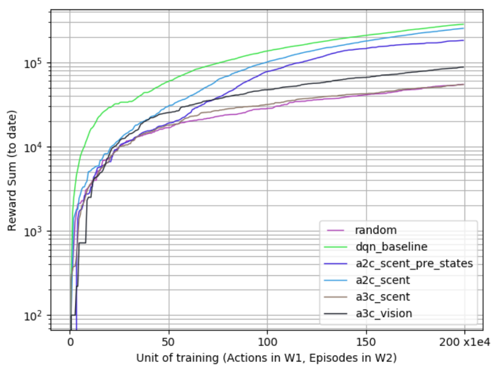
A2C Network The Actor Critic combines policy gradient and Q-learning in a way that the learning process becomes more quickly, and train for a never ending tasks. Though in this case, the variance is decreased, the bias introduced increase. In our implementation, the critic network has four fully connected layers and output dimension is 1, the actor network has 5 fully connected layers and output dimension is 3. We tried different inputs like scent perception, scent perception along with previous states and vision perception. However, the former two could learn but did not do better than baseline, and the network with vision input cannot converge at this moment. See the Fig. 5 for detailed results. Still, we wondered this might be some hyperparameter tuning part was not good

A3C Network Since A2C gets higher bias for the predictions because there is only one agent learning, the replay memories are highly related, the A3C Network allows us to use multiple agents to get asynchronous updates of the grid world in the same environment. In our implementation, we had 8 processes running for training and 1 process running for testing at the same time. The test agent retested at an interval of 20 seconds. The network has 3 fully connected layers, 1 LSTMCell layer, 1 critic linear layer and 1 actor linear layer. The network initialization takes xavier uniform. Unfortunately, the result from A3C network with both scent and vision input did not learn well at this moment.

DDPG Network We first want to try DDPG on this problem. Since it is a policy gradient algorithm, and it directly learns the policy we want instead of the q-value, which might help the agent to learn a more complex environment. The results of the network are completely bad, which showed that the agent did not learn anything. In this way, we didn't plot the reward curve for this method. The reason behind the bad results is that the DDPG is good at learning complex, continuous actions, not the discrete ones. Also, the environment is simple but requires many different objects to pick, which lead to the failure of modeling a continuous action space. For our implementation, we used scent perception as our first tryout, critic network has 4 fully connected layers and 1 tanh layer. The actor network has 4 fully connected layers and 1 softmax output. But it didn't work. So we totally gave up this method. We thought next time, if there are project like mountain cars, we can try this one.



(a) linear cumulative



(b) log cumulative

Figure 5: **Cumulative Plots of A2C and A3C**

5 Results

The metric for us to evaluate a policy is to compare the total reward received in in 2×10^6 steps from each method, and at the same time, the cumulative reward plot should always has the tendency to increase.

Our experiments shows that methods like first reward-reshaping (reward of picking up tongs is equivalent to getting jellybeans) based on the baseline, Double DQN with vision input, Double DQN with scent and vision concatenated input, dueling networks with scent input (with or without reward shaping) achieve higher rewards than the baseline,which is shown in the Fig.3 and Fig.4. These results are in accordance with my expectations. What we are unexpected is that A2C and A3C performs worse than the baseline. They cannot get converged. DDPG keeps outputing zero rewards during learning, which means in my implementation, DDPG could not learn discrete action space.

The following are some typical hyperparameters:

Double DQN

batch size	32
initial ϵ	0.5
final ϵ	0.05
burn in size	2000
replay memory size	10000
learning rate	0.0001

Dueling DQN

batch size	64
initial ϵ	0.5
final ϵ	0.05
burn in size	2000
replay memory size	10000
learning rate	0.0001

A2C

number of step	20
critic learning rate	0.0001
actor learning rate	0.001

A3C

batch size	32
tau τ	1.0
number of step	20
number processing	8
learning rate	0.0001

DDPG

batch size	64
burn in size	2000
memory size	10000
clearning rate	0.001
alearning rate	0.0001

6 Discussion

For the grid world navigation task, we fist applied the Double DQN as the baseline, then we tried Dueling Network and A2C Network. After that, we still tried something with the A3C Network and the DDPG Network. The results shows that the Double DQN and the Dueling Network can do an incredibly good job for only the scent perception. Still, adding some vision perception of current state also help a lot. We reckon this is because that multiple sensings can coordinate with each other's metrics and finally produce a better result. Unfortunately, all our tryouts for the previous observations failed. In our scenario, the previous states seemed to be useless when the agent is exploring an infinity grid world. We thought that previous observations may somehow obstruct the agent from exploring the newer regional grid world he/she perceived. When using the A2C, A3C and the DDPG, the results were not good at all. I reckon A2C is harder to do tine tuning, and though we've tried many hyperparameter settings or the network structure settings, we still wouldn't be able to produce a rather good result. For A3C, the asynchronous agent may reduce the training time for dividing the CPU process. However, treating all the agents as an equivalent balanced factor might not be a good idea in the grid world navigation scenario. As for the DDPG, the idea is good, though it's successful for the continuous actions instead of the discrete ones we are having here.

The limitation of our work is we didn't get enough time to implement more of our ideas and thoughts. Also, seen from the images we showed all above, they still got some improved space if we trained for longer steps. However, the plots and the discussions we showed here did provide some intuition in the ML models and the NEL environment. We're quite happy about this.

In fact, we want to find more possibilities of doing the data preprocessing. The most fascinating part is we want to create a mechanism that in each step, when we get the vision perception of the current grid world, we update the perception of the regional grid world by reshaping the reward of each object by their distances compared to the agent position. Say, I've got diamond for +100 reward, but if I should move 7 steps to get this, I'll reshape the reward into $+\frac{100}{7}$. In this way, we kind of providing with some intuition about the efficiency, and teach the agent to pick for something more valuable when they're easier to reach.

7 Reference

[1] Chen, Xi, et al. "Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments." arXiv preprint arXiv:1804.10500 (2018).

[2] Gnanasekaran, Abeynaya, Jordi Feliu Faba, and Jing An. "Reinforcement Learning in Pacman."

[3] Weber, Théophane, et al. "Imagination-augmented agents for deep reinforcement learning." arXiv preprint arXiv:1707.06203 (2017).

[4] He, He, et al. "Opponent modeling in deep reinforcement learning." International Conference on Machine Learning. 2016.

[5] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." AAAI. Vol. 2. 2016.

[6] Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." arXiv preprint arXiv:1511.06581 (2015).