# Navigation in Wild World

Haosen Xing, Xueqian Li

亲爱的，我爱你呢~~O(∩_∩)O

**Algorithm 1:** Baseline Double DQN

**Input:** *s*=Scent(or Vision)
**Ouput:** *Q*=Q-value
**begin**
  Initialize *s*;
  **while do**
    *a*=*ε-greedy* chosen from *s* using policy derived from $Q_{evaluate}$, $Q_{target}$
    Taken *a*, get next observation *s'*, *r*
    **if** *num_tong* == 0 and *Vision* == [0, 1, 0] then
      $r_{diamond}$=0
    **end**
    Replay memory for $\boxed{s, s', a, \text{and } r}$
    **while** *memory_size* > *burn_in_size* **do**
      Sample batch
      **If** *steps* < *target_replace* **then**
        Update $Q_{evaluate}$ using $\boxed{CNN}$
      end
      **if** *steps* % *target_replace* == 0 **then**
        Update $Q_{target}$
      **end**
    **end**
  **end**
  **return** *Optimal policy*
**end**

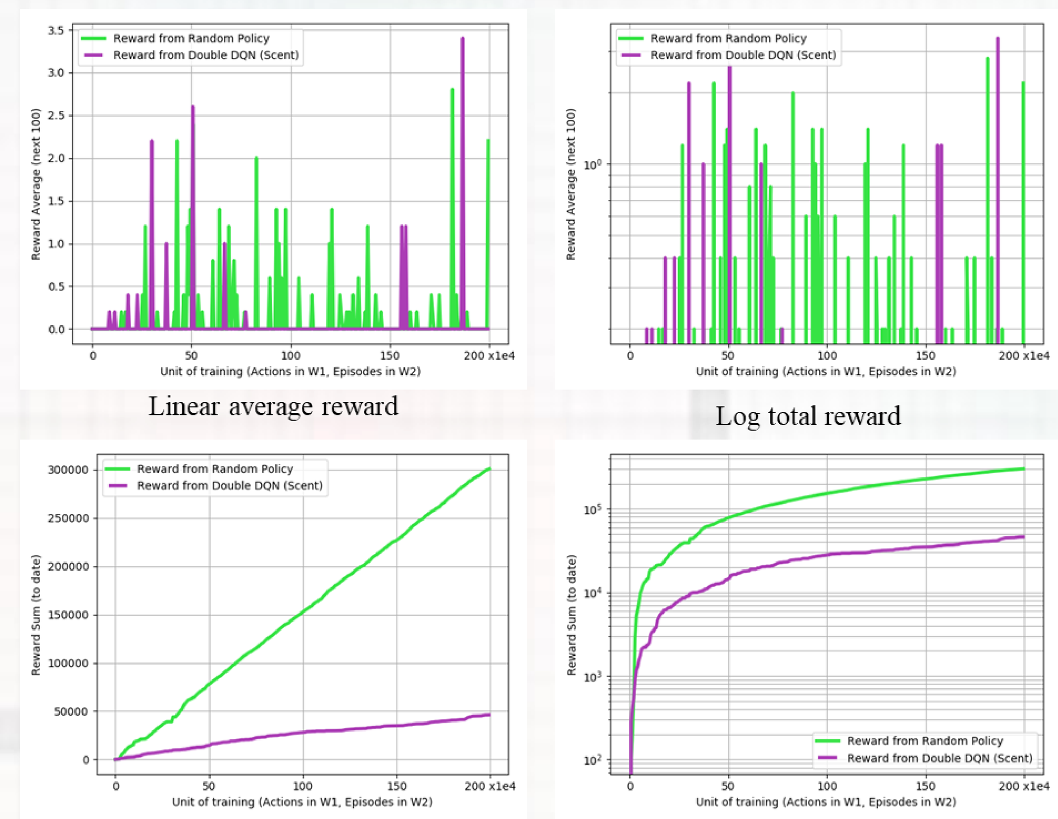
Scent → 3→32 → 32→64 → 64→32 → 32→3 (FCN)

Vision: 3→16 → 16→32 → 32→32 (Conv, 1d/2d) → 3872→512 → 512→128 → 128→32 → 32→3 (FCN)

Intuition:
1. Vision Network is larger.
2. Small layer size is sufficient for network regarding scent.
3. Double DQN is a sufficiently good method.

## Baseline

Approaching Method: *Double DQN*
Input: *Scent*
Reward Distribution: *Vision*
Network Structure: *4 FCN*



Linear average reward — Log total reward — Linear sum reward — Log sum reward

## Double DQN Contribution

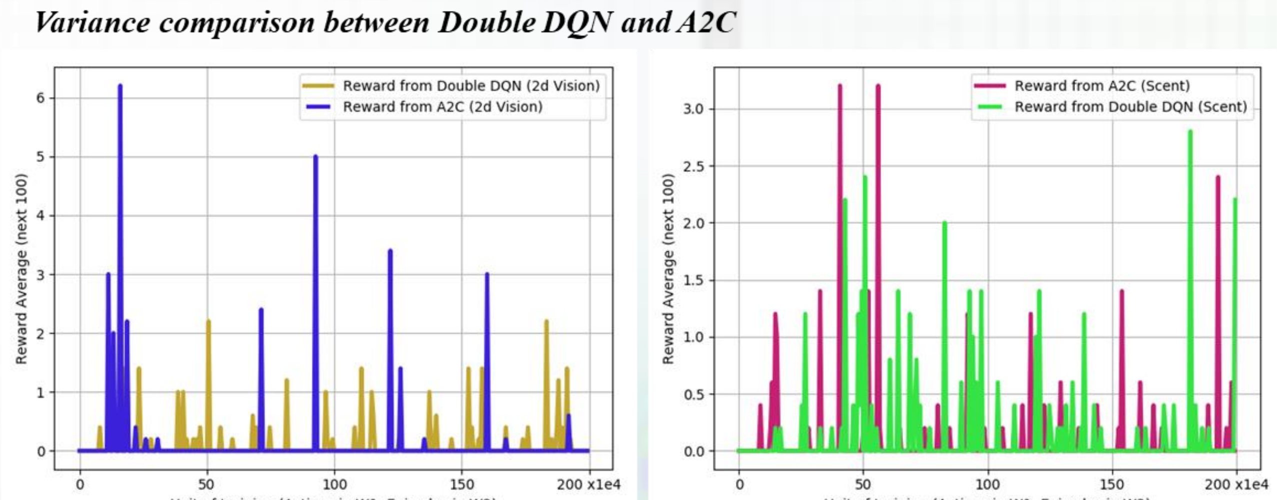
experience replay— remove correlations between samples

target Q network — avoid overestimating Q-values, more stable

The reward for baseline is obviously improved compared to the random policy.

**We are looking forward to find a method producing a better policy than the baseline.**

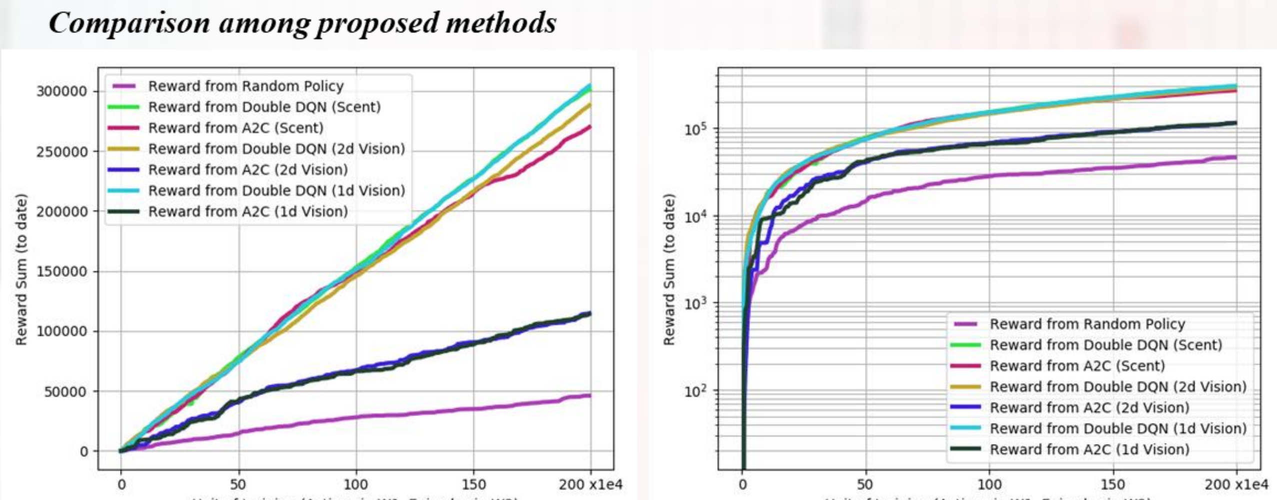
*Variance comparison between Double DQN and A2C*



Linear average reward for scent input — Log average reward for vision input

Intuition:
1. Vision might help here, but need further hyperparameter and network structure refinement (hidden layer size etc.).
2. 1d or 2d convolution seemed has no big difference here.
3. The *critic* network can be smaller than the *actor* network, but will the larger size *critic* network help? Or to use the identical network structure?

*Comparison among proposed methods*



Linear sum reward — Log sum reward

## Future

### *DDPG (Deep Deterministic Policy Gradient):*

uses evaluation networks and target networks in both critic and actor networks

experience replay

select action according to the current policy and exploration noise

### *PPO (Proximal Policy Optimization):*

based on A2C, uses an adaptive KL penalty to control the change of the policy at each iteration — compute an update that both minimizes the cost function and ensures the derivation from the previous policy small

### *A3C (Asynchronous Advantage Actor-Critic):*

based on A2C, uses multiple agents to explore the state space simultaneously — give uncorrelated updates to the gradients

Intuition:
1. In our navigation world, we have shown that both scent and vision input in A2C network experience larger variance in reward function.
2. We plan to use DDPG to bring experience replay into policy gradient in order to lower the variance.
3. With PPO, we plan to use KL function to control the step size which is also a potential way to lower the variance.
4. A2C with vision input doesn't do well in our navigation world, probably because the observations received by the agent are highly correlated. A3C could be a better method to decorrelate.
5. Also, we plan to try concatenating the FCN for scent and the Conv for vision to get better results.
6. We may add the *number of tong* as a network input to help the network learn better.

## *Summary*

*Compared to the baseline (Double DQN with scent input), our A2C with vision input (both 1d and 2d) perform worse than baseline and other trials are nearly equivalent to the baseline.*

***We haven't found a method that could beat the baseline.***

## Exploration

Approaching Method: *A2C*
Input: *Scent/Vision*
Reward Distribution: *Vision*
Network Structure: *4 FCN/1d or 2d Conv*

### A2C

***Pros:***

learns both Q-value (value-based) and π (policy-based), reduce the high variance

update at each step, converge faster than policy gradient

***Con:***

updates are correlated

### Policy Gradient RL

***Pros:***

easy to converge

can learn stochastic policies

effective in high-dimensional

***Con:***

converge to a local optimum

**Algorithm 2:** Advantage-Actor Critic

**Input:** *s*=Scent (or Vision)
**Ouput:** *Q*=Q-value
**begin**
  Start with *actor* model $\pi_\theta$ and critic model $V_\omega$
  Initialize *N*, *s*;
  **while do**
    Generate *N* steps $\boxed{S_0, A_0, r_0, \ldots, S_{N-1}, A_{N-1}, r_{N-1}}$ follwing $\pi_\theta$
    **for** *t* from *N-1* to 0 **do**
      $V_{end} = V_\omega(S_{t+N})$
      $R_t = \gamma^N V_{end} + \sum_{k=0}^{N-1} \gamma^k \, (r_{t+k} \; if \; (t+k<N) \; else \; 0)$
    **end**
    $\boxed{L(\theta)} = \frac{1}{T}\sum_0^{N-1}(R_t - V_\omega(S_t)) \, log\pi_\theta(A_t|S_t)$
    $\boxed{L(\omega)} = \frac{1}{T}\sum_0^{N-1}(R_t - V_\omega(S_t))^2$
    Update $\pi_\theta$ using $\nabla L(\theta)$
    Update $V_\omega$ using $\nabla L(\omega)$
  **end**
  **return** *Optimal policy*
**end**

Vision

actor: 3→16 → 16→32 → 32→32 (Conv) → 3872→512 → 512→128 → 128→32 → 32→3 (FCN)

critic: 3→16 → 16→16 (Conv) → 1936→512 → 512→128 → 128→32 → 32→1 (FCN)

Scent

Actor: 3→16 → 16→16 → 16→16 → 16→3

Critic: 3→128 → 128→1

FCN