3D Simulation for Robot Arm Control with Deep Q-Learning

10-20%	☑
20-40%	☑
40-60%	☐
60-80%	☐
80-100%	☐
Keywork	
URL	
備註	
論文性質	sim2real

Abstract

We also present preliminary results in direct transfer of policies over to a real robot, without any further training.

首個把影像手臂抓取，虛擬環境訓練模型，再**直接**移植到真實世界案例

Approach

Our approach uses a DQN [7] which approximates the optimal action-value (or Q) function for a given input:

$$Q^*(s,a) = \max_{\pi} E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a). \quad (1)$$

Here, $Q(s,a)$ defines the *q-value* of action a and state s. A *policy* π determines which action to take in each state, and is typically the action associated with the highest q-value for that state. $Q^*(s,a)$ is then the optimum policy, defined as that which has the greatest expected cumulative reward r when this policy is followed, over each time step k between the starting state at time t and some termination criteria (or infinite time). The discount factor λ gives weight to short-term rewards and ensures convergence of Equation 1.

The Q-learning update rule uses the following loss function at iteration i:

$$L_i(\theta_i) = E_{(s,a,r,s')\sim\mathcal{D}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right]^2, \quad (2)$$

最基礎的Q-learning，不是Nature或double-Q，另外有權重衰減值，loss是MSE

In this paper, we use this approach to train a 3D simulation of a 7-DOF robotic arm in a control task. On each iteration of training, an image is taken of the virtual environment and passed through the network to produce motor actions.

We define **an episode as 1000 iterations or until the task has been completed**. The task is considered completed when the arm **has grasped the cube and lifted it to a height of 30cm**. At the end of each episode, the scene is reset, and a new episode begins. We define the maximum number of iterations to be 1000 so that the agent does not waste time in episodes that cannot be completed successfully —for example,if the agent knocks or drops the cube out of its grasping range. We use an epsilon-greedy exploration approach which decides between a random action or the action corresponding the highest Q-value produced by the DQN. During training,epsilonis annealed over a period of 1 million episodes.

1.env's max mutual-value is 1000

2.the completed condition is lifted it to a height of 30cm.

3.epsilon 值，這裡不理解是退100萬次歸零還是其他方式

Our state is represented by an image of the scene which includes both the arm and cube. We define a set of 14 actions that control the gripper and 6 joints of the arm. Each joint can be controlled by 2 actions — one action to rotate the joint by

one degree clockwise, and another action to rotate one degree counter clockwise. The gripper follows similarly,with one action to open the gripper,and the other to close it. Our network,summarised in Figure 1,has 14 outputs corresponding to the Q-values for each action given the current state

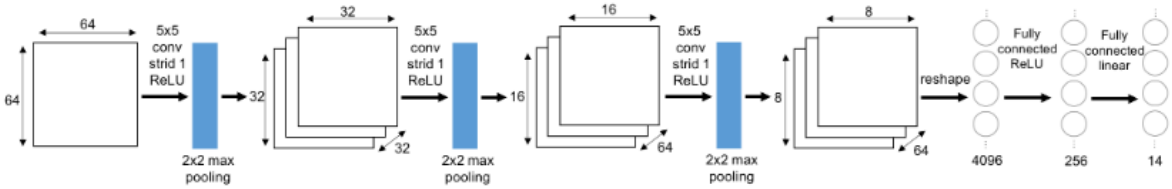
During each of our experiments,we fix the discount rate at = 0.99 and set our replay memory size to 500.000. Gradient descent is performed using the Adam optimisation algorithm [19] with the learning rate set at = 6 x 10(-6次方)

1.14個值，其中6個joint，1個夾爪閉合，標準的取max Q方法

2.衰減值 0.99

3.Adam

3.replay memory 500.000



3層卷積附maxpool，卷積stride為1。 後面接兩層全連結層，直到action count 14。

三種獎勵值

1.抓取積木至離地面30cm的高度，reward 100

2.抓取積木，遠離地點，愈高reward愈高

3.夾爪跟積木的距離，當距離躍進reward越大，最終reward為1。為何是自然數(e)沒講，但有說實驗時值會設定 r =0.25

Algorithm 1: Reward function

if *terminal* **then**
    $r = 100$
else if *target.grasped* **then**
    $r = 1 + target.position.y$
else
    $r = e^{-\gamma \times distance}$
end

Interestingly, we observed the arm returning to grasp the cube if it was dropped during the lifting motion (due to a random "open gripper" action selected via the epsilon-greedy method). This demonstrates the ability of the agent to recognise high rewards associated with grasping the object from a range of states, despite having never been trained for a re-grasping action.

因為貪婪值有時會隨機抓取，如果在抓取過程中不小心觸發放開的action，機器手臂會重新抓取，從這也認識到手臂對抓取物件，與高獎勵的有著認知關係。

看似沒必要，不過這樣提起來，想想的確是蠻有趣的...

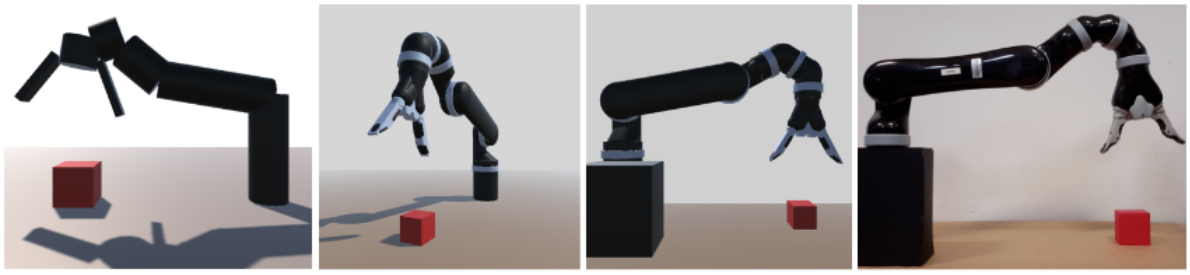
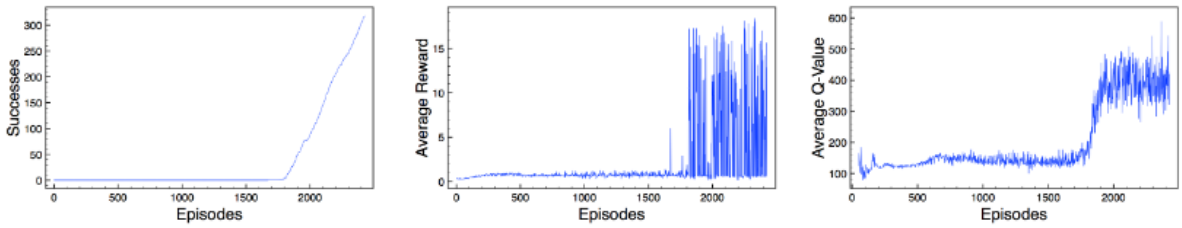


Figure 2: Image (a) shows the first version of our simulation. Both images (b) and (c) show the second version of our simulation that was used to train a policy that would be tested in a real-world environment (image (d)). A black box is placed in front of the base in order to avoid complicated modelling of the clamp holding the arm to the table, and the wires connected to the robot.

(a) 第一種版本， (b)、(c)為第種版本

看不出版本的差異，這邊也沒明確講怎調整第二種版本接近真實版本，不過眼前觀察，(a)的手臂的確是蠻暗的...



(a) Cumulative successes (b) Average Reward (c) Average Q-Value

(a)累積的成功次數 (b)平均獎勵 (c)平均Q值

感覺雖然成功，但a顯示的是個穩定的成功機率，非100%

1800 episode開始成功率明顯提昇。

Following this, we attempted to make our network learn to deal with a range of starting joint configurations and starting cube positions. At the start of each episode, the joint angles were set in a similar configuration to image (a) in Figure 2, but with variations of 20 degrees in each joint. During training,each time the agent was successful in completing the task, the cube was moved to a random

location with in a 200cm(2次方) rectangular area. A comparison of this expriment and the previous are summarised in Table 1. Here, we see the importance of training for tolerance to varying initial conditions compared to training with a fixed initial condition, where the success rate jumps from 2% to 52%.

1.手臂的自由度為20

2.積木位置隨機初現在平方200公尺內

Table 1: Success rate when running 50 episodes with $\epsilon = 0.1$. Environment A refers to keeping the joint angles and cube position constant at the start of each episode, while Agent A is the agent that was trained in this environment. Environment B refers to setting the joint angles and cube position randomly at the start of each episode, while Agent B is the agent that was trained in this environment.

Environment	Agent A	Agent B
A	56%	64%
B	2%	52%

訓練的初始條件如果是會變換的，則成功率可從2→52%，有點奇怪就是，B有隨機性，A沒有，當然落差會很大。也可能只是、以比較個差異的程度

Figure 4 shows a visualisation of the learned value function on a successful episode of 112 iterations (or frames). There are 5 images labelled from A to E which correspond to the frame numbers in the

graph. The frames from A to C show a steady increase in the Q-values which is a result of the agent getting closer to the cube. At frame C, the Q-values fluctuate briefly due to the network trying to determine whether or not it has grasped the cube. By the time we get to frame D, there has been a large jump in the Q-values where the agent seems certain that it has the cube. The Q-value then continues to rise as the height of the cube increases, and finally peaks as the agent expects to receive a large reward for task completion. The visualisation shows that the value function is able to evolve over time for a robot control task.

這過程很有趣，Max-value的值逐漸升高，這也表明TD是有效的，但延伸個問題是是否必須要至最後某個節點才能為max最大值？例如E滿足所有的獎勵條件，三種獎勵皆有拿

到，它如果是最後的獎勵條低值會如何?

另外這有點懷疑，B跟C是不是放錯位置了。

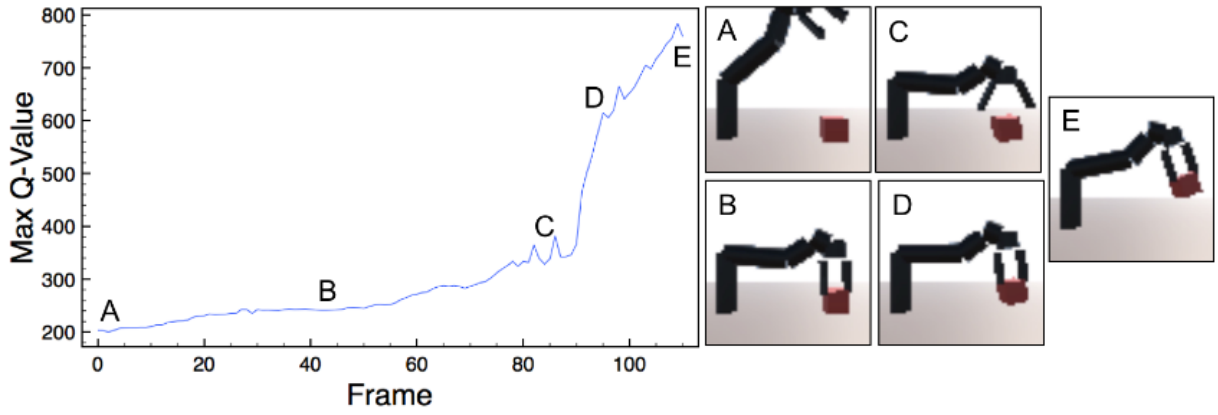


Figure 4: Visualisation of the learned value function on a successful episode of 112 iterations/frames.

To explore direct transfer to a real robot, we first needed to ensure that the simulated world resembled the real-world as much as possible. For this, we set out a scene in the real world and then created a more visually-realistic simulation than the first attempt. Images (c) and (d) in Figure 2 show both the real-world scene and its simulated replica. We took the network trained on this simulation and ran it directly on the real-world version of the arm with epsilon fixed at 0:1. As we had hoped, the agent exhibited similar behaviour to its simulated counterpart, moving the gripper directly towards the cube — suggesting that transferring from simulation to real-world is possible.

epsilon調整為0.1

嘗試虛擬model轉移到真實

However, the action to close the gripper was in fact rarely chosen by our trained policy when applied to the real-world, and so the agent was unable to fully complete the task in one sequence. Instead, the agent typically progresses towards the cube and remains in nearby states. In order to test if the agent could complete the task had it closed its gripper, we started the agent in a position where it has the cube in its grasp. We re-ran the experiment from this starting

state, and then the agent completed the task successfully by lifting the cube in an upwards direction. This illustrates that transferring control of the gripper is more challenging than that of the 6 joints on the arm, perhaps due to there being less room for error in a binary "open or close" action, compared to an action involving movement in continuous 3D space.

移動接近可以，但抓起來無法，後來用手動調整讓手臂抓積木，接著它是可升起往上的。作者推測是因為夾取的sample不多，大部分都是sample的採樣。不過這也無法解釋那位什麼虛擬的時候沒出現這問題，猜也可能有些機構或電機上的干擾因素

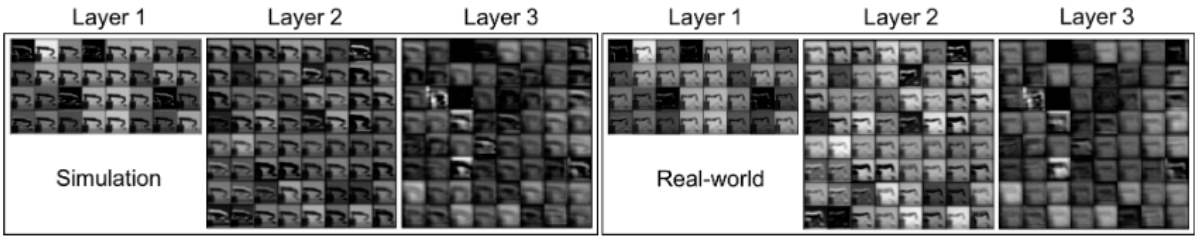


Figure 5: Comparison of the feature map activations of the simulation and real-world. The feature map activations on the left are from simulation while the activations on the right are from the real-world.

圖表示虛擬跟現實，在卷積層中間的結構輸出，東西長差不多。