QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation

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

1. 怎決定放置物品的?
2. 多個動作怎決定執行哪個?
3. 有講初始化的事嗎?
4. 隨機策略怎決定

底部物品高度：-0.1677

夾爪初始高度：0.47629

夾爪首部位移：0.3112

夾爪初始高度：0.182

夾爪最終低度：0.073

操作 -0.1 實際位移距離： 0.183 → 0.110

_dv = 0.06

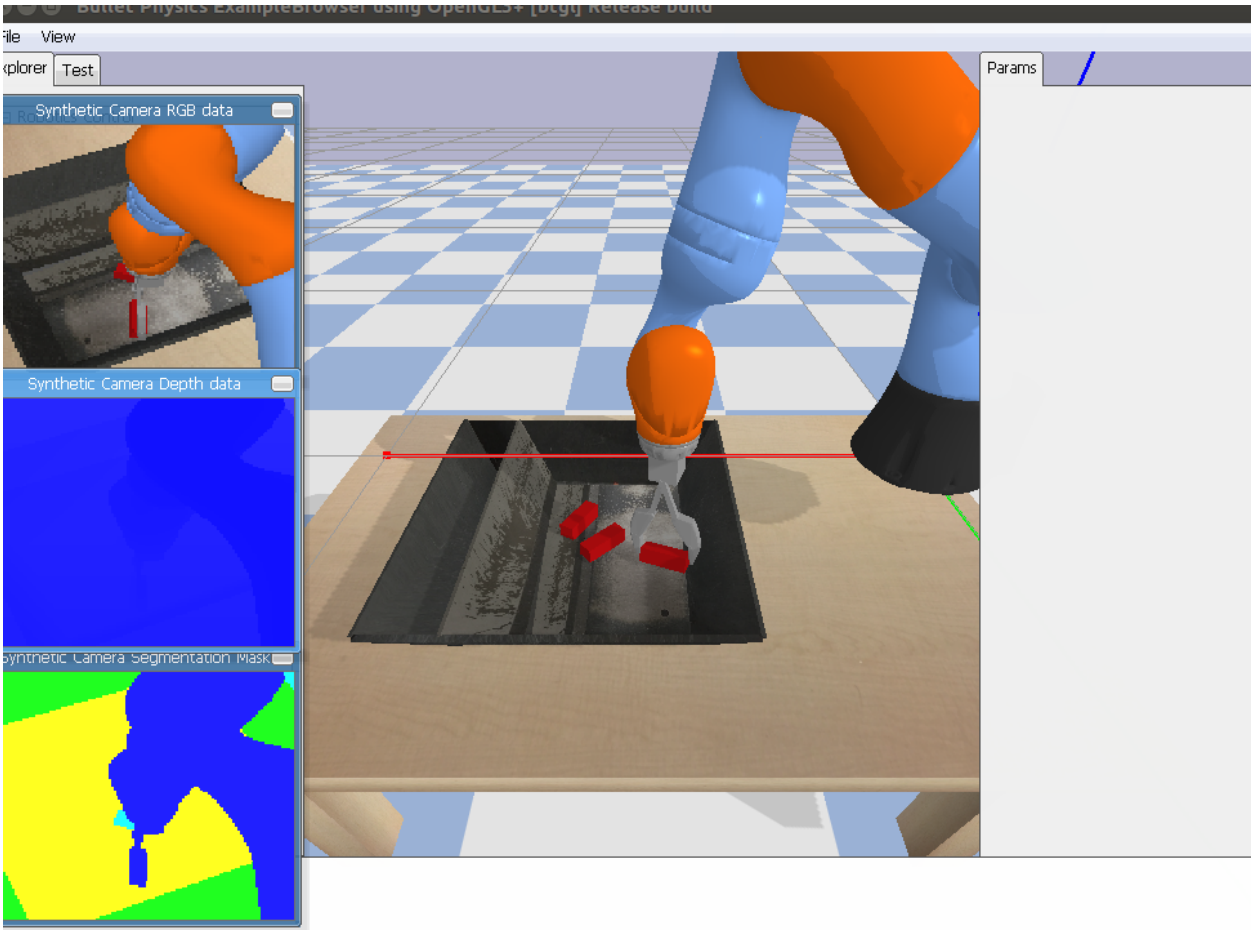
判斷是否結束：0.1

- 解決指定高度問題，物品獎勵高度無法一致
- ~~夾爪過於到下方無法閉合 (用time_step解決，超過5則往上)~~
- 一開始高度過高，直接返回 (應該是隨機高度動作問題)
- ~~獎勵有2的問題~~
- cross_entropy append the discrete action，在CEM裡面可以再優化
- random_sample_box 因為gin設定不釐清，暫時不動
- ~~在高度0.13時，開始降緩速度的下降程度~~
- 碰到某些物品夾爪會卡住
- 可以改從圖像轉成物品的位置，先訓練看看
- map 跟 batch順序影響 https://bre.is/AhvUVLJp
- 可以新增一個，曾達到底部的狀態，與完成此狀態可得0.1獎勵，並設計如果碰到底部則給懲罰
- time_step 改由env提供
- **scale time_step**
- **改變相機角度**

將研究

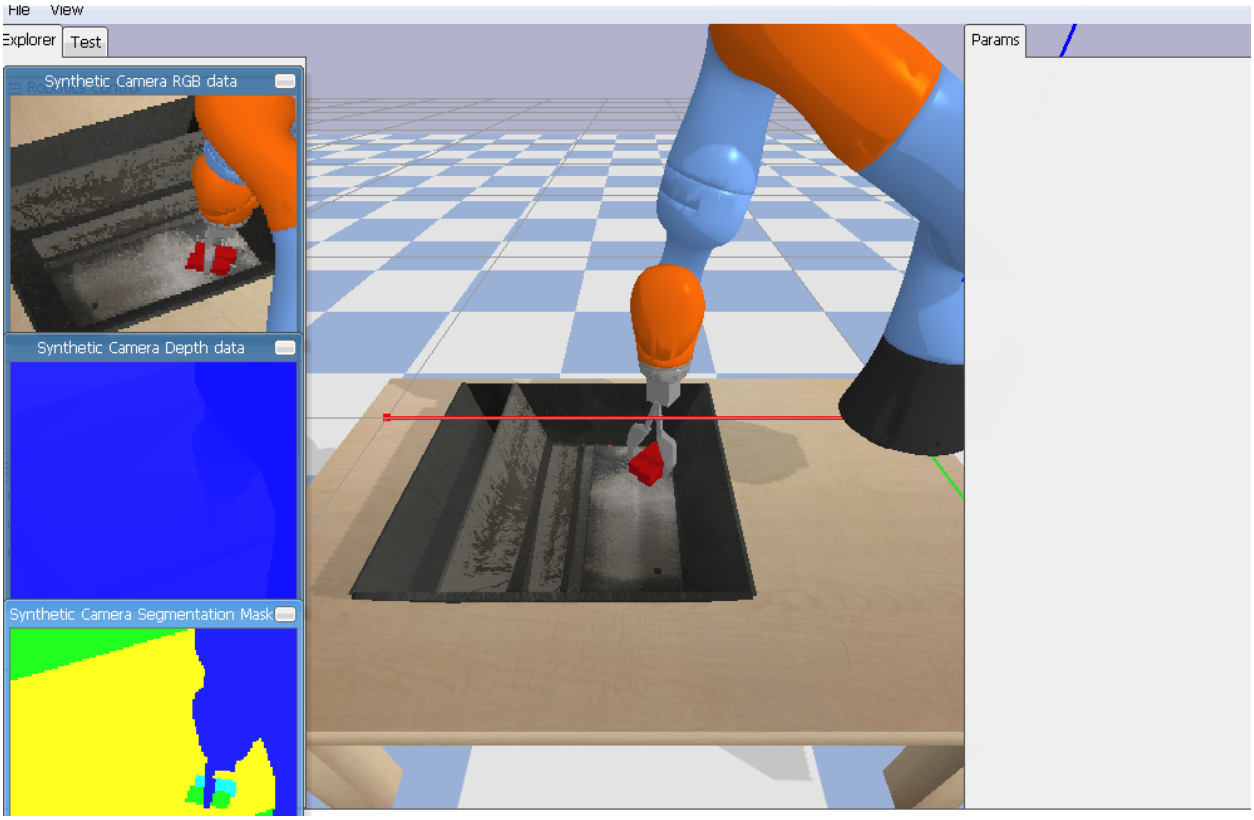
- batch_normal layer
- replay queue
- 隨機object 位置初始縮放 & 抓取位置縮放
- sample_object position train
- CNN DNN改造

陽性樣本卡住



- 卡牆或底部，或許要加懲罰項，對照DQL過於樂觀可以確定不會有保守狀況出現。而且目前已經對機器的底部卡住實作了

夾兩個，不知道曷不起來



技術債

Aa file	method	content
policies.py	sample_fn	action_size -3
q_graph.py	random_sample_box	num_samples-3(數量), action_size -3(動作)
Untitled		

先不要切換，訓練一百萬次

fine joint turn

5 Dynamic Vision-Based Grasping

Reward function. The reward is 1 at the end of the episode if the gripper contains an object and is above a certain height, and 0 otherwise. Success is determined

by using a background subtraction test after dropping the picked object, as discussed in Appendix D.4. Note that this type of delayed and sparse reward function is generally quite challenging for reinforcement learning systems, but it is also the most practical reward function for automated self-supervision. To encourage the robot to grasp more quickly, we also provide a small penalty r(st;at) =0:05 for all time steps prior to termination, when the model either emits the termination action or exceeds the maximum number of time steps (20). This penalty may in principle result in target values outside of[0;1], though we found empirically that this does not happen.

成功得1，每次步伐懲罰0.05，最高步伐至20，很少有機會達到滿懲罰的狀況，應該是手臂提早符合條件e了。

B Exploration and Dataset Bootstrapping

As is standard in Q-learning, we evaluate using one policy (evaluation policy) and collect training data with a different policy (exploration policy). Our evaluation policy π-eval chooses each action by maximizing the Q-function value using our QT-Opt algorithm described in Section 4.2. For data collection, we used two different exploration policies π-scripted, π-noisy at different stages of training.

採最大值做評估：π-eval　探索：π-scripted, π-noisy

During the early stages of training, a policy that takes random actions would achieve reward too rarely to learn from, since the grasping task is a multi-stage problem and reward is sparse. This was indeed what we observed during early experimentation. For this reason, we collect our initial data for training using a scripted policy π-scripted that successfully grasps 15-30% of the time. The scripted simplifies the multi-step exploration of the problem by randomly choosing an (x ; y)coordinate above the table, lowering the open gripper to table level in a **few random descent steps**, closing the gripper, then returning to the original height in a few ascent steps.

為了應對多步驟動作的獎勵稀疏問題，採取π-scripted策略蒐集樣本。選擇隨機的(x y)座標，在幾步內下降打開的夾爪往下，接著關閉，然後上升至原本高度。

We compared initial data collection with π-scripted vs. initial data collection with π-prior, a grasping model based on Levine et al. [27]. In our real-world experiments, we used π-prior, but in a simulated comparison, we found that initializing training with either policy leads to the same final performance and data efficiency. The data generated by either policy has similar distributional properties, as discussed in Appendix C.1, which is sufficient to bootstrap learning.

During the later stages of training, we switch to data collection with π-noisy. This exploration policy uses epsilon-greedy exploration to trade off between choosing exploration actions or actions that maximize the Q-function estimate. The policy π-noisy chooses a random action with probability = 20%, otherwise the greedy action is chosen. To choose a random action, π-noisy samples a pose change **t, r** from a Gaussian with probability 75%, a toggle gripper action **g-open g-close** with probability 17%, and an episode termination **e** with probability 8%.

探索機率值為20%，其餘則採貪婪算法。其中有75%是隨機移動，打開與關起為17%，終止為8%。不過沒提到探索值得衰減為多少。

C Simulated Experiments: Dataset Size, Off-Policy Training, MDP Design

State	Termination action	Intermediate reward	Discount factor	Perf. at 300K steps	Perf. at 1M steps
Image+gripper status+height	No	-0.05	0.9	75%	95%
	No	0	0.9	68%	92%
	No	0	0.7	50%	90%
Image only	No	-0.05	0.9	25%	81%
Image+gripper status+height	Yes	-0.05	0.9	67%	94%

Table 7: Simulation studies for tuning grasping task parameters

The results in Table 7 show that richer state representation results in faster convergence and better final performance. A small reward penalty does much better than decreasing the discount factor. Finally, giving the policy greater

control over the episode termination yields performance on par with the engineered termination condition.

懲罰比增加遞減好， 增加termination action比沒增加不好一點

We put special focus on the termination action: by letting the policy decide when to terminate, the policy has more potential to keep trying a grasp until it is confident the grasp is stable. We argue that explicitly learning a termination action would be beneficial for many manipulation tasks, since it makes the design of the MDP easier and it forces the policy to truly understand the goal of the task.

但增加termination action可增加時間效率，與實質讓算法了解目標。

Figure 10: Performance graphs of simulation studies for tuning grasping task parameters

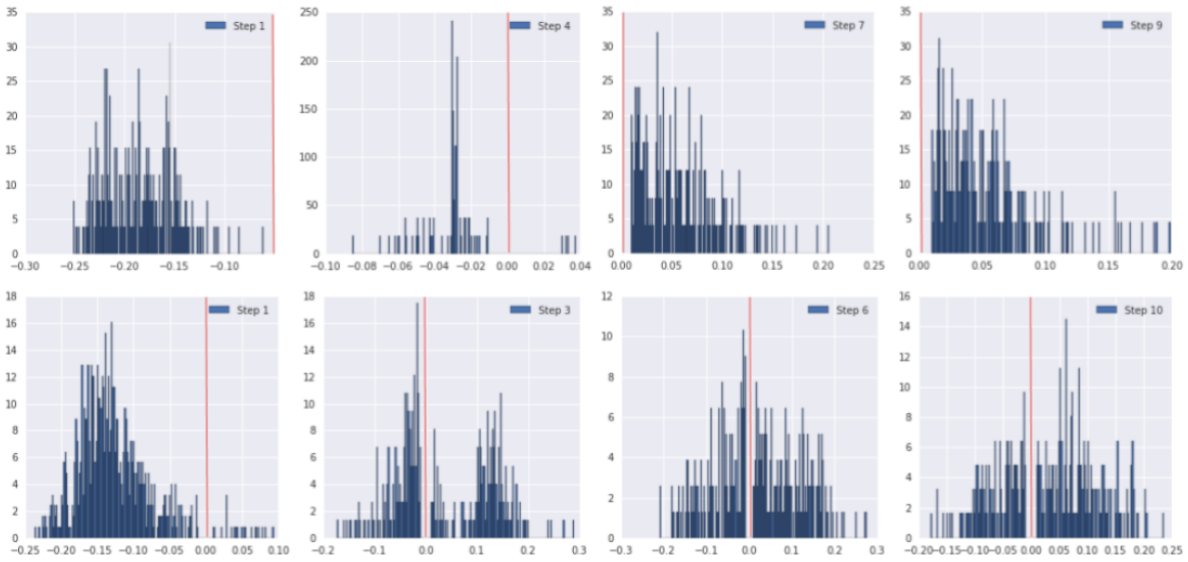


Figure 11:(top row) The distribution of z-coordinate of the pose translation of actions selected by the randomized scripted policy. The z-coordinate values are biased towards negative values during the first few steps on the descent phase of the episode, then biased towards positive values at later steps on the ascend phase. Such a biased action distribution over states provides poor action space exploration and is insufficient to learn a good Q-Function. (bottom row) The z-coordinate of actions selected by a suboptimal QT-Opt policy. The distribution is roughly centered around zero (red axis) at each step providing good action space exploration.

下面是更優學的policy，因為z軸由子優化policy學習？難道同時有多個QT-Opt policy嗎?

Data efficiency Interestingly, our algorithm is more data efficient than the supervised learning based algorithm from Levine et al. [27] work, achieving higher grasp success with fewer robots continuously generating training data (see Table 8).

Name	Sim Robots	Success
QT-Opt (ours)	30	88%
	60	95%
Levine et al. [27]	60	55%
	280	71%
	1000	85%

Table 8: Data efficiency comparison in simulation.

We argue that the algorithm from Levine et al. [27] is less data efficient because it optimizes a proxy objective of 1-step classification accuracy, which values all data points equally. Our QT-Opt policy values data points based on how they influence reward. This focuses optimization on pivotal decision points that are very important to get right, such as learning when to close the gripper. Doing so lets the model optimize grasp success more efficiently.

跟Levine et al 論文相比，對於每個點都是平等價值，QT-Opt更注重於關鍵點的學習(決定夾取點)

D Grasping MDP: State Space, Action Space, and Reward Evaluation

The goal in designing the MDP is to provide a framework in which an agent may learn the necessary hand-eye coordination to reach, grasp, and lift objects successfully. Our sensors include a 640×512 RGB camera and joint position sensors in arm and gripper. In the following we describe representation of state, action and reward and discuss the stopping criterion of our grasping task.

D.1 Observation Space

To provide image observations It that capture maximal information about the state of the environment, we mounted the camera to the shoulder of the robot overlooking the entire workspace (see Fig. 2). In practice, we also observed that

our agent was able to learn more effectively when observations also include some proprioceptive state. Specifically, this state includes a binary open/closed indicator of gripper aperture and the scalar height of the gripper above the bottom of the tray. The full observation is be defined as st = (I ; g aperture; g height). The model-input It is a 472×472 crop of the full-size image with random crop anchor. This way the model is discouraged from relying on a rigid camera-to-base transformation which cannot be assumed across several robots. Instead, the policy is forced to learn about the presence and the view of the gripper and impact of the actions in the environment, which is critical for the emergence of closed-loop self-corrective behaviours. We also apply image augmentation. The brightness, contrast, and saturation are adjusted by sampling uniformly from[0:125;0:125];[0:5;1:5], and[0:5;1:5]respectively. To stay consistent, the same augmentation is applied to the images in the current states and next states 0 . Random cropping and image augmentation is only done at train time. At inference time, no augmentation is used and we always crop to the center 472×472 square of the image.e

學習圖片資訊，包含夾爪有無合起以及手臂的高度。圖片是經過隨機裁切，這是因應多台實體機器拍攝會有不同的狀況影響。以及做了亮度、對比與飽含度的隨機調整。為了維持一致性，當下與下一張照片會使用相同的資料增加。檢測的時候不做任何資料增強，影像會從中心點 472×472裁切。

D.2 Action Space

The agent's action comprises a gripper pose displacement, and an open/close command. The gripper pose displacement is a difference between the current pose and the desired pose in Cartesian space, encoded as translation t t R 3 , and vertical rotation encoded via a sine-cosine encoding rt R 2 . A gripper open/close command is encoded as one-hot vector[g close;t;g open;t] {0;1}2 . In addition, as this is an episodic task with a finite horizon, our agent has to decide when to terminate. For a baseline policy, we implement a heuristic stopping criterion that triggers when the arm is holding an object above a threshold height. Such a heuristic is task-specific and might bias learning in a sub-optimal way. Thus, we introduce an additional action component et which allows our policy to learn a stopping criterion and decide when to terminate autonomously. The full action is defined as at= (tt ; rt ; g close ; t ; g open ; t ; e t).

a = (t 移動位置、r 旋轉軸、g-close 閉合、g-open 開合 one-hot 介於{0, 1}, e 自主停止}

D.3 Reward Function

The agent is given a reward of 1 at the end of an episode if it has successfully grasped an object from the bin. In all other cases the reward is 0. In order to detect a successful grasp we use an image subtraction test, see Fig. 12. An image of the scene is captured after the grasp attempt with the arm moved out of camera view. Then a second image is taken after attempting to drop the grasped object into the bin. If no object was grasped, these two images would be identical. Otherwise, if an object was picked up, certain pixels in the two images will be different. A grasp is labeled success if the image subtraction is above a threshold.

在放置箱子中檢測兩者的差異性，符合一個閥值則判定為抓取成功，獎勵為1

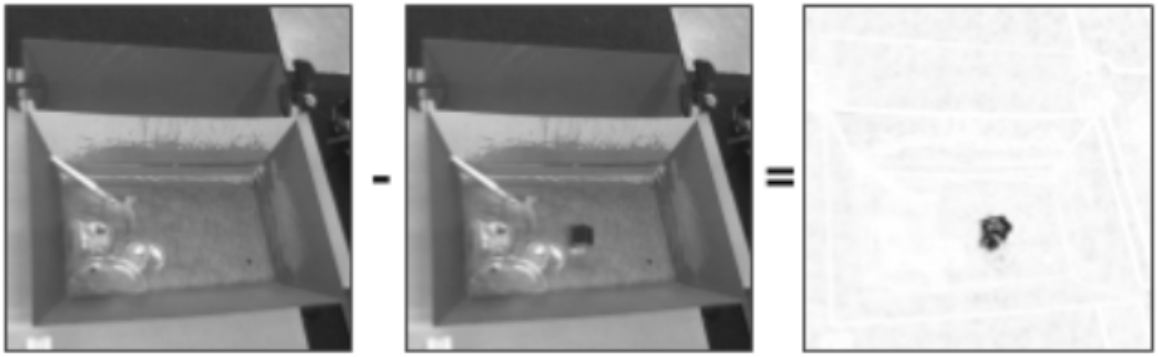


Figure 12: Grasp success is determined by subtracting images before an object is dropped into the bin (left) and after it was dropped (right).

Because we use a simple image subtraction test to compute the reward, in practice the labeling is not perfect. We analyzed the failed grasps from the model that reaches 96% grasp success, and discovered that out of 28 misgrasps, 3 were actually successful grasps that have been classified improperly by our image subtraction test. Our background subtraction test might not detect small objects, making it vulnerable to false negative registrations. This shows that our learning

algorithm can train a very good policy even with a small noise in the reward function.

從96%成功率中，有28次誤抓，3次其實成功，但卻標記成失敗。小物品無法使用此方法減去(猜測可能是多個物品的時候，被比較大的擋到)。但就算有錯誤樣本，模型依然訓練很好，維持抗噪性。

D.4 Grasp execution and termination condition

Algorithm 1 Grasping control-loop

Pick a policy $policy$.
Initialize a $robot$.
while $step < N$ and not $terminate\_episode$ **do**
  $\mathbf{s}$ = $robot$.CaptureState()
  $\mathbf{a}$ = $policy$.SelectAction($\mathbf{s}$)
  $robot$.ExecuteCommand($\mathbf{a}$)
  $terminate\_episode$ = e {Termination action e is either learned or decided heuristically.}
  r = $robot$.ReceiveReward()
  emit($\mathbf{s}$, $\mathbf{a}$, r)
end while

With state, action and reward defined in the sections above, we now describe the control-loop executed on our robots. As described in Algorithm 1 we control our manipulator by closing a loop around visual input and gripper commands. At the end of an episode we detect grasp success and returnthe sequenceof state actionpairs to be fed back into the learning pipeline or stored offline. Depending on the executedpolicy, our termination actioneis either learned or decided heuristically as we discuss below.

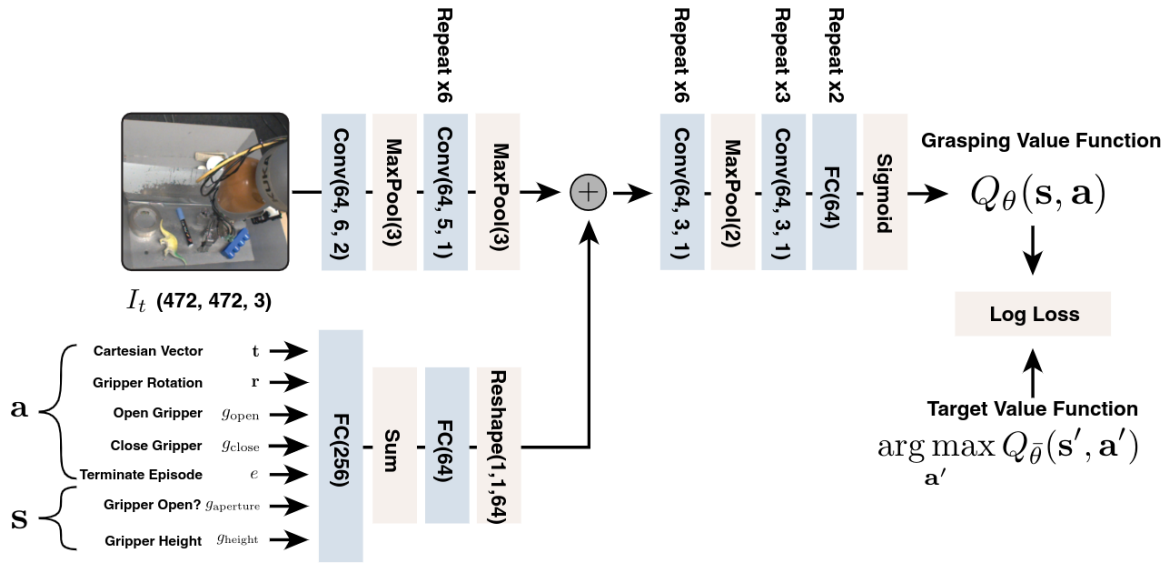


Figure 13:The architecture of grasping Q-Function. The input image is processed by a stack of convolutional layers before the introduction of the action and vector-valued state variables (gripper status and height). These are processed by several fully connected layers, tiled over the width and height dimension of the convolutional map, and added to it. The resulting convolutional map is further processed by a number of convolutional layers until the output. The output is gated through a sigmoid, such that our Q-values are always in the range[0;1].

Scripted termination condition Initial experiments used a heuristic termination condition to de-termine when the policy is done. The heuristic detects when the gripper is holding an object, and the policy is continuing to command upward actions after it has passed a height threshold.

Algorithm 2 Scripted termination condition

```
termination_height = 0.13
s = robot.GetState()
gripper_height = s.height_of_gripper
gripper_closed = s.gripper_status='CLOSED'
a = robot.GetAction(s)
next_action_height = a.translation.z
if gripper_closed and gripper_height > termination_height and next_action_height > gripper_height then
    terminate = True
else
    terminate = False
end if
```

Learned termination condition In principle, a policy may use the camera image to determine if it has successfully grasped and raised an object. It may inform such knowledge through a discrete termination action. To learn the termination action, the reward structure must be slightly changed, reward= 1 is assigned when grasp success=True and gripper_termination_height > threshold. This enforces the policy to indicate episode termination only after the gripper lifts the object, which makes it easier to provide visual feedback for robust grasps; objects will fall out of a brittle grasp during the lifting process, and terminating the grasp after the object falls gives richer negative examples.

只有在抓取成功與達到一定高度才會有獎勵，也有可能抓不好，因終止抓取但掉落，提供了負面樣本

有個細節，z是移動高度還是絕對高度?

the gripper aperture and distance between the gripper and the floor. Next, we studied discount factors and reward penalties. Finally, we compared a scripted episode stopping criterion to a learned termination action. In these experiments, we use 60 simulated robots, using $\pi_{scripted}$ policy for the first 120K gradient update steps, then switching to the π_{noisy} policy with $\epsilon = 0.2$. These exploration policies are explained in Appendix B. The grasp performance is evaluated continuously and concurrently as training proceeds by running the π_{eval} policy on 100 separate simulated robots and aggregating 700 grasps per policy for each model checkpoint.

使用scripted policy在60個模擬手臂上，做12萬次的訓練。

然後切noisy policy，機率值為0.2，100個手臂，分別產出700次抓取， 7w

為了避免過擬合，在新的on-policy中， 在前面1M 次，把分數從1%→50%

Real on-policy data is generated by 7 KUKA LBR IIWA robot arms where the weights of the policy $Q_{\bar{\theta}_1}(s, a)$ are updated every 10 minutes. Compared to the offline dataset, the rate of on-policy data production is much lower and the data has less visual diversity. However, the on-policy data also contains real-world interactions that illustrate the faults in the current policy. To avoid overfitting to the initially scarce on-policy data, we gradually ramp up the fraction of on-policy data from 1% to 50% over the first 1M gradient update steps of joint finetuning training.

mutil step rl

Joint Finetuning

Each convolution and fully-connected layer uses batch normalization [41] and the ReLU nonlinearity. The value function handles mixtures of discrete actions (open, close, terminate) and continuous actions (Cartesian vector and gripper rotation), making the same state-action space amenable to learning future tasks like placing and stacking. In Table 2, we demonstrate empirically that the model performs better when provided with redundant state representations (g_{aperture}, g_{height}). All weights are initialized with truncated normal random variables ($\sigma = .01$) and L2-regularized with a coefficient of $7\text{e}{-5}$. Models are trained with SGD with momentum, using learning rate 0.0001 and momentum 0.9.