# Accelerate Training of Reinforcement Learning Agent by Utilization of Current and Previous Experience

Chenxing Li[1,3], Yinlong Liu[2], Zhenshan Bing[2], Fabian Schreier[1,3], Jan Seyler[3] and Shahram Eivazi[1,3]
[1]*University of Tübingen, Tübingen, Germany*
[2]*Technical University of Munich, Munich, Germany*
[3]*Festo, Esslingen, Germany*

Keywords: Q-function Targets Via Optimization, Data Efficiency, Hindsight Goals Techniques, Offline Data Collection, Dynamic Buffer.

Abstract: In this paper, we examine three extensions to the Q-function Targets via Optimization (QT-Opt) algorithm and empirically studies their effects on training time over complex robotic tasks. The vanilla QT-Opt algorithm requires lots of offline data (several months with multiple robots) for training which is hard to collect in practice. To bridge the gap between basic reinforcement learning research and real world robotic applications, first we propose to use hindsight goals techniques (Hindsight Experience Replay, Hindsight Goal Generation) and Energy-Based Prioritization (EBP) to increase data efficiency in reinforcement learning. Then, an efficient offline data collection method using PD control method and dynamic buffer are proposed. Our experiments show that both data collection and training the agent for a robotic grasping task takes about one day only, besides, the learning performance maintains high level (80% successful rate). This work serves as a step towards accelerating the training of reinforcement learning for complex real world robotics tasks.

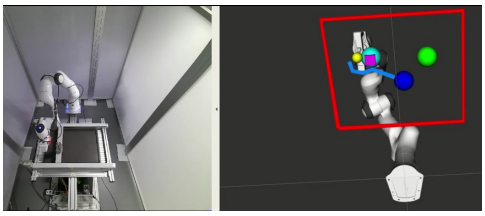
## 1 INTRODUCTION

In recent years, reinforcement learning (RL) algorithm become one of the most popular artificial intelligence techniques. In various famous human-machine competitions such as StarCraft competitions between AlphaStar and MaNa (Vinyals et al., 2019), the machine behaviours performed very well with help of RL. As a general learning method, it can be used for different tasks and the agent learns the policy via exploring action space on its own simulator (Zhang et al., 2015) not like hard coded control methods where any change to environment may lead to error states such as it could make the robot be stuck in some poses without adaptive control design. Normally RL performs well more on low dimension space, for continuous tasks such as robotics manipulation, however, a long training time is necessary to reach reliable learning level (Sutton and Barto, 2018). To provide reliable performance under environment changes, the mapping from inputs to outputs in RL can be substituted with a neural network for updating (Li, 2017). Based on this mapping suppose, several techniques in RL are proposed to improve the learning performance of agent such as Deep Q-learning (DQN) (Szepesvári, 2010), Double Q-learning (Hasselt, 2010) and (Van Hasselt et al., 2016) and Deep Deterministic Policy Gradient (DDPG) (Wiering and Van Otterlo, 2012). Apart from that, Hindsight Experience Replay (HER) (Andrychowicz et al., 2017), Hindsight Goal Generation (HGG) (Ren et al., 2019) and Energy-Based Prioritization (EBP) (Zhao and Tresp, 2018) are proposed by researchers previously to address the problems of efficient data sampling for real world application.

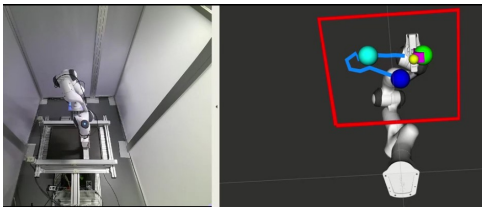
Google X team proposed Q-function Targets via Optimization (QT-Opt) as a scalable deep reinforcement learning algorithm for vision-based robotic tasks (Kalashnikov et al., 2018). It can deal with the actions which contain continuous and discrete action space based on the Clipped Double Q-learning (Fujimoto et al., 2018) with cross-entropy method (CEM) that can avoid direct max operation over Q-values. Besides, it can overcome the notoriously unstable problem caused by policy network and Qt-opt combines both advantages of on-policy learning and off-policy learning (Bodnar et al., 2019). In practice, QT-Opt needs numerous computing resources as well as time and hardware to collect offline data such as Google had used 7 robots over the course



Li, C., Liu, Y., Bing, Z., Schreier, F., Seyler, J. and Eivazi, S.
Accelerate Training of Reinforcement Learning Agent by Utilization of Current and Previous Experience.
DOI: 10.5220/0011745600003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 698-705
ISBN: 978-989-758-623-1; ISSN: 2184-433X



(a) The robot arm first moved behind the object in Pushing task.



(b) The robot arm finally pushed the object to the desired position successfully.

Figure 1: The original QT-Opt methods are already implemented in real world tasks. The left scenarios of both figures are the real world Pushing task showing that the robot arm is pushing the object to the desired position. The right scenes of both figures are synchronous to the left real world motions and show the trajectory of the robot arm end-effector from the starting point (blue sphere) to the final position. The light blue sphere presents the original object position, the pink one is the current object position, the small yellow sphere shows the current position of end-effector and the green one is the desired goal position. The robot arm first moved behind the object, then pushed the object to desired goal position successfully.

of several weeks to collect the offline data. With a distributed robotic grasping experiment they needed to keep the environment information same across all robots which make it unpractical. Here we aim to use relatively much less resources (e.g. 14k CPU core vs. 32 CPU core) while achieving a competitive learning performance. As such, we proposed several techniques to improve the learning performance and tested them in simulation using 7 degrees of freedom robotic arm (Franka Emika robot arm). Our paper makes the following contributions:

- An PD control data generation is proposed to make the offline data collection simple and efficient.
- HER, HGG and EBP are adapted to the QT-opt structure for efficient data sampling.
- A dynamic buffer mechanism is proposed to reduce the size of offline data as well as generation of sub-optimal trajectories in offline data.

Besides, Reaching task and Pushing task are already implemented in the real world based on the original ideas with QT-Opt while Google X team did not present them. Like Figure 1 shown, the robot arm pushed the object to desired goal position successfully. The aim of implementing them is to extend the Google work and verifies the feasibility of our proposed methods so as to build the theoretical and experimental fundamentals for complex tasks. The focus of this paper is the grasping task (PickAndPlace task). In this paper, we adapted the QT-Opt to non-distributed QT-Opt for quick comparison. Based on own version of QT-Opt, the performance comparison over grasping task are given in simulation. In the future, the grasping task in the real world can be implemented like the accomplished tasks. The paper presents the introduction and preliminaries containing the contributions and the fundamentals of RL techniques. Then the methods that used in the research are discussed in the section methodology. Section results and discussion analyse the results. Finally, conclusions and future work plan are given in the following sections.

## 2 PRELIMINARIES

Reinforcement learning is one of main learning branches of machine learning (Hammersley, 2013). The mechanism of RL is based on Markov decision processes (MDP) (Garcia and Rachelson, 2013) and pursue an optimal policy (Puterman, 1990). In this process, at each time step $t$, the agent in one state $s_t$ performs an action $a_t$. Criteria to generate the action is based on the policy $\pi(a_t \mid s_t)$. In every step, agent receives a reward $r_t$ according to the reward function $r(r_t \mid s_t, a_t)$ and then transitions to the next state $s_{t+1}$ according to the transition function $p(s_{t+1} \mid s_t, a_t)$. An optimal policy $\pi^\star$ can be derived by maximizing the expected discounted sum of the rewards like Equation 1. The discounting factor $\gamma$ is limited in the $[0,1]$ for avoiding the infinity rewards for the environments without final states (Bennett and Hauser, 2013).

$$\pi^\star = \arg\max_\pi \mathbb{E}_{r\sim p(\tau|\pi)}[\sum_{t\geq 0} \gamma^t r_t] \tag{1}$$

Based on that, Q-function can be used to find the optimal policy. After setting environments based on consecutive states, the Q-function is written as Bellman equation as followed (Barron and Ishii, 1989):

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r\sim p(\tau|\pi)}[r_t + \gamma Q^\pi(s_{t+1}, a_{t+1})] \tag{2}$$

Further, neural network can be integrated with reinforcement learning (DQN). The responding target value is:

$$Q_t = r(s, a, s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{3}$$

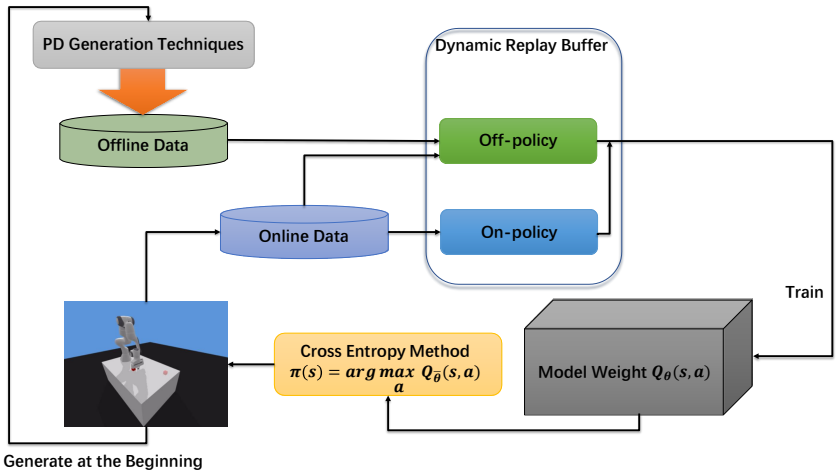



Figure 2: Modified method structure. The offline data can be generated via PD controller mode, then the online data are collected. A dynamic buffer is used to choose the data for updating. Finally, CEM chooses the responding policy based on the neural network weights. Rather than original QT-Opt, Bellman updater is substituted with the new buffer mechanism and the offline data is collected by PD generation techniques.

Algorithm 1: Cross Entropy Method (CEM).

**Inputs:** *Neural network outputs and Gaussian parameters*

**Outputs:** *Actions and Q-values*

1: Initialize standard error and mean based on a Gaussian distribution
2: **for** $i = 1, 2, \ldots,$ until the iteration you choose **do**
3: Generate N actions based on a Gaussian distribution
4: Obtain the Q-values in neural networks
5: Choose the best M samples from N actions
6: Update the CEM parameters with the M samples
7: **end for**
8: **return** maximum actions and the responding Q-values

However, if the target is susceptible to error, then the maximum that we get is generally larger than the true maximum and it can lead to overestimation bias. Then the values continue to propagate through the Bellman equation. Finally, the agent may learn in the wrong direction. The max operation is the main reason of overoptimistic value estimates. To solve this, one can decouple the selection from the evaluation. The Clipped Double Q-learning is based on this idea to update the policy, while two independent estimates of the true Q value are included. To avoid the overestimation of Q-values, the minimum of the two next-state action values produced by two Q neural networks is used to compute the updated targets (Hasselt, 2010).

The traditional policy can be updated by getting the maximum of Q value, then its responding action in robotic environments is the policy in each steps. Normally, a second network for updating Q-values or actions is needed. Thus, a cross-entropy method (CEM) is proposed to perform the optimization avoiding the need for a second maximizer network. The Algorithm 1 shows the process of CEM (Kalashnikov et al., 2018). With CEM and clipped Double Q-learning, the Q-values can be updated as:

$$Q^{\star}(s_t, a_t) = r_{t+1} + \gamma min_{i=1,2,\ldots} Q_{\theta_i}(s_{t+1}, a_{CEM}) \quad (4)$$

The above equation is the core of pure QT-Opt algorithm which is an on-policy training algorithm. It is meaningful to extend this algorithm to a structure which combines both advantages of on-policy and off-policy learning. The agent can learn fast in the right direction when it explores enough behaviours like Figure 2 shown. Apart from that, how to generate a good offline data is vital concern for the QT-Opt algorithm. From the paper (Kalashnikov et al., 2018), offline data can be obtained by adopting the previous online data. The difference of on-policy and off-policy learning is whether updating based on the current strategy. In this status, the online data in current updating is on-policy, while for the next updating, these data is off-policy. Besides, the data that we obtained need to include enough successful data, otherwise, it always learns the bad behaviour which is not wanted. Thus, they first bootstrapped the data with QT-Opt, then they saved all data and constructed the data with enough successful data and relative complete action space.

Another well-known problem with RL is data efficiency. The reinforcement learning can generate

the data via the simulator on its own not like deep learning, while at the beginning, the generated data is not so efficient as to the agent to learn the correct behaviour. Hindsight Experience Replay (HER) and Hindsight Goal Generation (HGG) are the methods proposed to improve the task performance via increasing the data efficiency. HER can replay each episode with a different goal which the agent achieved and it can construct a number of achieved hindsight goals based on the intermediate states (Andrychowicz et al., 2017). Apart from that, HGG generates valuable hindsight goals which are easy for an agent to achieve in the short term and are also potential for guiding the agent to reach the actual goal in the long term (Ren et al., 2019).

EBP is a method to choose valuable data in buffer considering energy changes. In robot grasping task, the object kinematic and potential energies changes in trajectory are considered as the criteria of valuable data which means that the object was caught and moved to other place. If the energy changes of the trajectory are higher, the replay buffer has higher chance to choose this trajectory for updating (Zhao and Tresp, 2018). Then the agent can use less time to fully learn the behaviour based on valuable data in buffer.

## 3 METHODOLOGY

In this paper, we focus on how to use relatively less resources (a day of training and a gaming PC with RTX2070) to accomplish the competitive learning performance like Google X team did in 2018 when they used QT-Opt to accomplish 96% over robot grasping task which took several months for training and collecting data. To avoid the need for large scale cluster we use the physical information as observation in this paper rather than images. Our methods can be be formed with three perspectives. The Figure 2 shows the method structure and contains the following points.

- Adapting HER, HGG, and EBP to the QT-Opt structure.
- A PD control data generation method.
- A dynamic buffer mechanism.

First, HER and HGG can help learning with increasing data efficiency. How to use HER with QT-Opt has been proposed before (Fujita et al., 2020) in large scale cluster (not the focus of this paper) but the combination of QT-Opt and HGG which has not been studied yet. In this paper, EBP is used to increase the data efficiency and can be combined with HER

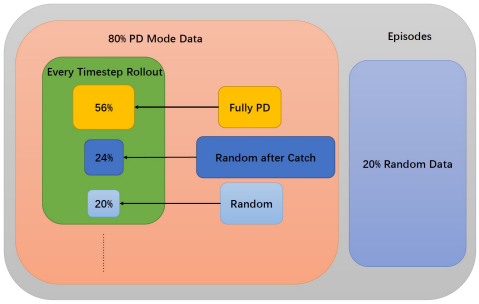



Figure 3: PD Method to Generate Offline Data.

and HGG. HER needs to replay the episode with the hindsight goals, so many successful trajectories can be created in learning. As such, there is no need to change the QT-Opt structure.

HGG needs to choose the goal and initial state before each rollout. The fundamental of HGG is based on the following suppose:

$$|V^{\pi}(s,g) - V^{\pi}(s',g')| \leq L \cdot d((s,g),(s',g')) \quad (5)$$

Here, $V^{\pi}$ shows the responding policy, the function $d(\cdot)$ is a metric that $d = c||\phi(s) - \phi(s')||_2 + ||g - g'||_2$. $c$ is the hyperparameter which is larger than zero to make sure trade-off between the distances between the initial states and the final states. $\phi(\cdot)$ is a state abstraction to map from state space to goal space. Based on Equation 5, we can rewrite the equation via $(s,g) \sim T$, $(s',g') \sim T$ and solve the optimization problem:

$$\min_{\pi,T}[L \cdot D(T,T^{\star}) - V^{\pi}(T)] = \min_{\pi,T}[c||\phi(s) - \phi(s')||_2 + ||g - g'||_2 - \frac{1}{L}V^{\pi}(T)] \quad (6)$$

Where $D(\cdot)$ is the Wasserstein distance based on $d(\cdot)$. We assume that $T$ is hindsight. Then $K$ particles will be adopted from $T^{\star}$, instead of dealing with $T^{\star}$, the approximation method is used to simplify the tasks, hence, for every task $(\hat{s}_0^i, \hat{g}^i) \in \hat{T}^{\star}$, the hindsight trajectory $T$ should minimize the following sum:

$$\sum \omega((\hat{s}_0^i, \hat{g}^i), \tau^i) \quad (7)$$

where we define $\omega((\hat{s}_0^i, \hat{g}^i), \tau^i)$ is transformed based on Equation 7. Then the corresponding achieved state will construct the hindsight goals in $T$.

From the above deduction (Ren et al., 2019), we can find that HGG needs to make several changes to adjust QT-Opt structure. The hindsight goals $T$ are stored in a pool mechanism. The updated Q-values for HGG need to obtained by the inputs computing in pool which we called *AchievedPool* via neural network. Normally, algorithms like DDPG have a

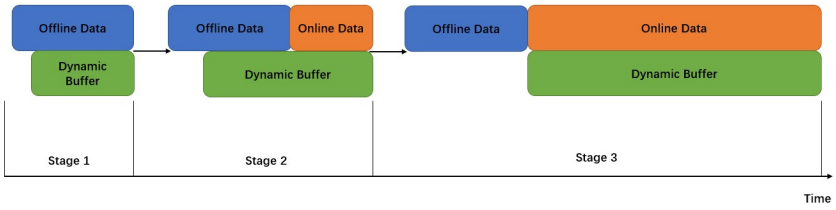



Figure 4: Dynamic Buffer Mechanism. At the beginning, the rest $3/4$ data are considered as offline data in QT-Opt. With online data adding, the offline data buffer contains both offline data and online data for off-policy updating. Finally, the buffer will be full of online data. In this period, the buffer is dynamic with the online data adding.

specific network to generate these Q-values. However, QT-Opt uses CEM algorithm which will generate actions and Q-values in same networks. The achieved Q-value will be obtained by CEM with current weights from $V^{\pi}(T)$, CEM is also the mechanism of actions generation in Qt-opt, so we use the responding Q-values outputs of CEM to Equation 6.

Second, a new offline data bootstrapping technique is proposed. It is known that QT-Opt can take advantages of off-policy and on-policy learning when training more complex tasks. Simple randomly generated data can not directly be used as the offline data in QT-Opt structure as proposed by authors (Kalashnikov et al., 2018) and the QT-Opt need at least 30 percents successful data. It is difficult to make sure that the random data can have at least 30 percents successful samples even if we get the data from on-policy training. Thus, an efficient method to generate enough valuable data is our concern.

We propose to use PD control method to generate offline data. PD control method is a simple method to control the agent to move fast and stable to our desired goal. $P$ is proportional parameter and $D$ is derivative coefficient, while $P$ can help fast achieving goal and $D$ can make the response stable. When we use the PD control method, for example in the PickAndPlace task, we design the robot arm first moves about the object and finger is open, the end-effector goes down to contain the object and further closes the finger to catch the object, finally, it catches the object to the goal area.

Our successful samples index among all samples can be reached. However, it is also not helpful to set all the offline data as successful data because then the agent can not learn which behaviour is not correct. For example, in grasping task, if the offline data is always successful, the finger could close when it is near the threshold of the object and catch the object, but the agent can never learn how to react if the finger is open in the above circumstance. To include unsuccessful examples, a random policy needs to combine with the PD control data. Besides, the random actions in PickAndPlace task are not often to present the finger with grasping the object and in this transport, the design to show that it drops the object or it starts fully random motions to test which angle is best to catch the object need to be also included. For the trajectories generated by our method, they are not fully from the PD control mode while they need some bad behaviours to let the agent learn not to behave that. Thus, bootstrapping in PickAndPlace task, we set at each time-step, there exists 56% probabilities where trajectories are fully PD control generated, 20% probabilities where trajectories are fully random mode without any control, the rest 24% probabilities where trajectories are integrated in PD control mode but with random motion in several steps as Figure 3 shown.

Third, we propose a dynamic buffer mechanism to further increase the learning performance (Figure 4). The PD generated data can be considered as $ScriptedTrajectories$ which are the trajectories which use successful policy to generate. Apart from PD control data generation, it also needs to add more explored policies in QT-Opt structure. The others are the trajectories which are based on exploration to generate which is the $ExploredTrajectories$. They also need to have at least 30% successful data (Kalashnikov et al., 2018) which can not be certain via fully random sampling.

Our PD controller method has the enough successful data while it does not explore all the policies. It has more data directly towards to the goal even with random scripted data. The explored policies that we need are sub-optimal policies. We can use the previous on-policy data as $ExploreTrajectories$ but it needs to satisfy the rate of successful trajectories, thus the robot needs to explore almost all action space to collect the data from Google X team.

To solve this problem, we propose to first use PD offline data for training while saving the transitions in each time step so we can add them to our database later. In our experiment, our data use the combination of these two policies, they are PD control data as offline data and following explored data as online data. With online data adding, the offline data buffer con-

tains both offline data and online data for off-policy updating. Finally, the agent can sample data fully on explored data. This is the method we called the dynamic buffer. Besides, this mechanism also avoids the Bellman updater in original QT-Opt structure for dynamic buffer not only generates *ExploreTrajectories* but also provides the agent which data need to be trained for updating network weights.

This technique can certain that the agent is able to explore the policies, and also have acceptable successful rate. In details like Figure 4 shown, for a buffer data with offline data and online data, first with one probability $\alpha$ we used energy sampling method, and with $1-\alpha$ percentage the data is sampled from the rest $3/4$ trajectories randomly. $3/4$ trajectories can be updated when adding more new data. At the beginning, these trajectories contain the enough offline data, and the offline data let the agent learn in the right way. Then the online data can be further included as *ExploredTrajectories* based on neural networks updating. This innovative technique makes sure the offline data is made of the *ScriptedTrajectories* and *ExploredTrajectories*, besides, the at least 30% successful data can also be certain. As such, the rely on offline data can be less because the more online data with the help of hindsight goals techniques like HER and HGG are more available to use. Furthermore, HGG technique can also generate more couples including the initial states and goals which are easy to reach. Then more virtual goals can be among the real goals, these new generated online data therefore become more potentials.

## 4 RESULTS AND DISCUSSION

The experiment environment (Gallouédec et al., 2021) is based on Pybullet engine (Coumans and Bai, 2021) and Gym (Brockman et al., 2016) with Panda robot from Franka Emika company. Before complex tasks, Reaching task and Pushing task are implemented to prove the feasibility of original QT-Opt in fully continuous space. In our simulation task, a continuous action space is used which means both robot joint control, as well as the gripper finger open and close operations are continuous. The non-distributed QT-Opt is implemented under the mentioned setup. For the typical robot PickAndPlace task, several comparison experiments are implemented. To evaluate the performance of our algorithm, we adopt the following measures. First, the successful rate is our main criteria to judge whether the method that we propose is effective or not. Normally, the *x-axis* is the episode information. One episode in our experiment is one complete test that the robot arm starts from the initial state and implements to reach goal position under maximum steps. The responding *y-axis* presents the information of successful rate. The successful rate is obtained after training one episode via implementing 50 tests without updating based on current neural network weights. Second, for several new methods that we proposed, one curve to present the efficiency is not persuasive. Hence, it is necessary to have a number of tests over this new method based on the same environment, then the mean value can be considered as the rational value for comparisons.

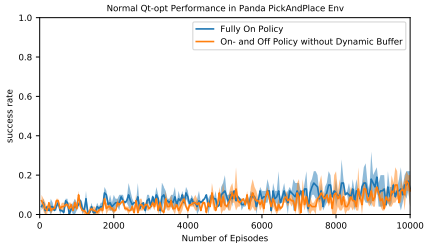



Figure 5: Normal QT-Opt Learning Performance. The blue curve shows the fully online Qt-opt learning while the orange one shows the Qt-opt combining with normal buffer using on-and offline data.

First, pure on-policy QT-Opt and QT-Opt with normal replay buffer (our baseline) are implemented over PickAndPlace task. The results can be seen in Figure 5. With normal buffer, or fully online data, QT-Opt performs bad learning performance over robot PickAndPlace task. Therefore, it is necessary to explore how to use the offline data to update the target value and improve the data efficiency.

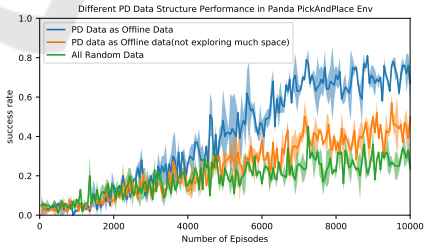



Figure 6: Different PD data learning performance with dynamic buffer. The blue curve is combined with HER, EBP and dynamic buffer based on PD offline data generated from our method. The orange curve in this figure shows the similar PD mechanism without sampling at each time-step rather maintain the responding fraction in whole episodes. The green curve is fully random bootstrapping to collect the offline data with dynamic buffer.

Further, a PD offline data generation comparison is implemented in PickAndPlace task (Figure 6). The blue curve is combined with HER, EBP and dy-

namic buffer based on PD offline data generated from our method. The orange curve in this figure shows the similar bootstrapping mechanism. However, this mechanism is not generating the data with the proposed percentages in each time step rather distribute in whole episodes. Besides, it contains only grasping status without opening fingers in PD episodes. This method can obtain good and fast successful data in buffer, but not explore the enough action space. And the green curve is fully random bootstrapping to collect the offline data with dynamic buffer. From that, we can conclude that our PD offline data generation can explore much action space and achieve better learning performance than other techniques. Our method can reach about 80% successful rate while the other two perform under 60% successful rate finally. Besides, it is necessary to include dropping the object circumstances when grasping successfully based on the comparison between orange curve and blue curve so that the agent can deal with the behaviours.

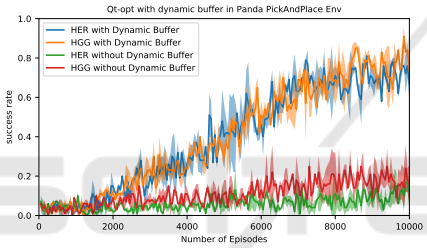



Figure 7: QT-Opt with hindsight goal techniques and dynamic buffer. The blue curve shows the performance of mixed-policy QT-Opt with HER and dynamic buffer, the orange curve presents how the mixed-policy QT-Opt with HGG and dynamic buffer performs. The red curve shows the circumstance of mixed-policy QT-Opt with HGG and normal buffer. While the green one shows the performance of mixed-policy QT-Opt with HER and normal buffer. All offline data length is 30000 steps, the sampling rate among the online data and offline data is 80%.

From the curves over QT-Opt with normal offline data in PickAndPlace task in Figure 5 which means that *ScriptedTrajectories*, the learning successful rate is limited, the data does not explore all behaviours in action space. As discussed in last section, *ExploredTrajectories* are needed. Hence, dynamic buffer is implemented and compared with the full PD control method generation data as offline data in the Figure 6 and 7. Comparing to the orange curve in Figure 5 as our baseline with normal buffer, the dynamic buffer version like the blue curve shown in the Figure 6 and 7 performs much better.

Apart from that, HGG and HER can be combined with QT-Opt and improve dynamic buffer learning effect for they can generate better data for next replay in replay buffer and make more midpoints to final success. Besides, HGG combination can achieve more successful rate and perform more stable than HER with QT-Opt in the final stage as Figure 7 shown, though the difference is not so significant (5% - 10%).

Further, different length of offline data learning performance are implemented as Figure 8. With more offline data, the learning performance shows faster and better learning over robot grasping task. With 30000 steps offline data, the learning successful rate reaches 75% while with 10000 steps offline data, the performance is about 55%.

With the help of dynamic buffer mechanism and hindsight goals techniques, the offline data size is around 30000 steps data which occupy less than 1k memories in PC. The vanilla QT-Opt on the other hand needed 580k memories data (Kalashnikov et al., 2018).

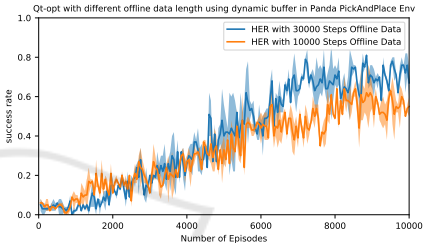



Figure 8: The learning performance with different number of offline data. The blue one shows the learning performance of mixed-policy Qt-opt with 30000 steps offline data. The orange curve has only 10000 steps offline data. The algorithm is based on HER and dynamic buffer. The sampling rate among the online data and offline data is 80%.

## 5 CONCLUSIONS

In this paper, modified QT-Opt as the basic method for efficient learning in robotics tasks is used to avoid overestimation of Q-values and be stable. PD control data generation with several random steps in data structure can construct offline data to accomplish better training in QT-Opt structure instead of collecting the large amount of previous data as offline data which needs more time and real world robots. Further, the agent can inspire from the limited numbers of control behaviours, imitate and explore the whole policy. The dynamic buffer mechanism combines the *ScriptedTrajectories* with *ExploredTrajectories* without manual insertion. Thus, the complexity of the collection data can be reduced significantly and the learning performance can be improved with hindsight goals techniques further. This novel combination shows competitive performance for efficient reinforcement learning and pro-

vides new research direction for QT-Opt learning over real world robot tasks.

## 6 FUTURE WORK

In the future, we plan to use a hybrid action space including both continuous joint control and discrete finger move. Based on current results, we expect to have the significant improvement in hybrid action space as well. In engineering, we also plan to mix simulation and real robot, then the neural networks weight can be transferred to real robot using transfer learning techniques (Torrey and Shavlik, 2010). We expect that the agent learn faster in further training in real world compare to start training without simulation data.

## REFERENCES


Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. *arXiv preprint arXiv:1707.01495*.

Barron, E. and Ishii, H. (1989). The bellman equation for minimizing the maximum cost. *Nonlinear Analysis: Theory, Methods & Applications*, 13(9):1067–1090.

Bennett, C. C. and Hauser, K. (2013). Artificial intelligence framework for simulating clinical decision-making: A markov decision process approach. *Artificial intelligence in medicine*, 57(1):9–19.

Bodnar, C., Li, A., Hausman, K., Pastor, P., and Kalakrishnan, M. (2019). Quantile qt-opt for risk-aware vision-based robotic grasping. *arXiv preprint arXiv:1910.02787*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Coumans, E. and Bai, Y. (2016–2021). Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org .

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477.

Fujita, Y., Uenishi, K., Ummadisingu, A., Nagarajan, P., Masuda, S., and Castro, M. Y. (2020). Distributed reinforcement learning of targeted grasping with active vision for mobile manipulators. *arXiv preprint arXiv:2007.08082*.

Gallouédec, Q., Cazin, N., Dellandréa, E., and Chen, L. (2021). Multi-goal reinforcement learning environments for simulated franka emika panda robot.

Garcia, F. and Rachelson, E. (2013). Markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 1–38.

Hammersley, J. (2013). *Monte carlo methods*. Springer Science & Business Media.

Hasselt, H. (2010). Double q-learning. *Advances in neural information processing systems*, 23:2613–2621.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Ren, Z., Dong, K., Zhou, Y., Liu, Q., and Peng, J. (2019). Exploration via hindsight goal generation. *arXiv preprint arXiv:1906.04279*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Torrey, L. and Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Wiering, M. A. and Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, learning, and optimization*, 12(3).

Zhang, F., Leitner, J., Milford, M., Upcroft, B., and Corke, P. (2015). Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*.

Zhao, R. and Tresp, V. (2018). Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, pages 113–122. PMLR.