Learning Robot Grasping from a Random Pile with Deep Q-Learning

Bin Chen[1,2], Jianhua Su[1,2(✉)], Lili Wang[1,2], and Qipeng Gu[1,2]

[1] University of Chinese Academy of Sciences,
Beijing 100190, People's Republic of China
[2] State Key Lab. of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Science,
Beijing 100190, People's Republic of China
jianhua.su@ia.ac.cn

Abstract. Grasping from a random pile is a great challenging application for robots. Most deep reinforcement learning-based methods focus on grasping of a single object. This paper proposes a novel structure for robot grasping from a pile with deep Q-learning, where each robot action is determined by the result of its current step and the next n steps. In the learning structure, a convolution neural network is employed to extract the target position, and a full connection network is applied to calculate the Q value of the grasping action. The former network is a pre-trained network and the latter one is a critical network structure. Moreover, we deal with the "reality gap" from the deep Q-learning policy learned in simulated environments to the real-world by large-scale simulation data and small-scale real data.

Keywords: Grasping · Deep Q-learning · Simulation to real

1 Introduction

The ability of robots to reliably grasp a wide range of novel objects can benefit applications in logistics, manufacturing, and housing robot. Traditional methods try to obtain the full knowledge of the model and pose of an object to be grasped, and then select the grasping configuration on the surface of the object. Contrary to the analytic approaches using the geometric information, learning-based methods pay more attention on the feature extraction, the pose estimation, and the object classification, etc. The resulting data is then used to retrieve grasps from some knowledge base or samples and rank them by comparison to existing grasp experience. Those works started to be popular in recent ten years.

Some researchers imitate human grasps to deal with the grasping problem. Based on the statistical modelling, statistical learning technology such as Gauss

This work was supported in part by NSFC under Grant number 91848109, supported by Beijing Natural Science Foundation under Grant number L201019, and major scientific and technological innovation projects in Shandong Province Grant number 2019JZZY010430.

X.-J. Liu et al. (Eds.): ICIRA 2021, LNAI 13014, pp. 142–152, 2021.
https://doi.org/10.1007/978-3-030-89098-8_14

process regression model (GPR) [1], Hidden Markov Models (HMMs) [2], Support Vector Machine (SVM) [3], neural network [4], have been employed to deal with imitation-based grasping problem. A comprehensive review work on this area can be referred to[5].

Some researchers regard a grasp as a problem from "RGB image" to "grasping configuration", thus the end-to-end deep learning approaches have been developed for grasping planning. Pinto [6] trained a Convolutional Neural Network (CNN) for the task of predicting grasp locations without severe overfitting. They recast the regression problem to an 18-way binary classification over image patches. Mahler et al. [7] trained Grasp Quality Convolutional Neural Networks (GQ-CNNs) using synthetic depth images, and then combined them to plan grasps for objects in the point cloud.

The deep learning-based methods inevitably require a considerable initial training data, which is tediously labelled by a human supervisor. However, it is reasonable to expect the robot can perform a new task more simply. The method is very challenging in practice. Some works show deep reinforcement learning (Deep RL) has achieved greet success in domains ranging from games [8,9] to robotic control [10], and perform well in the field of robotic manipulation, such as pushing [11], grasping [12] etc. Popov et al. [13] discussed the object grasping and precise stacking based on Deep RL. They provided a simulation study that investigated the possibility of learning manipulation end-to-end with Deep Deterministic Policy Gradient (DDPG) [14]. But their works are limited to single objects, and only give the simulation result. Some other works also have presented autonomous data collections through trial and error [2] or in simulated scenes [15], however, the policies trained on synthetic data may have reduced performance on a physical robot due to inherent differences between models and real-world systems. This simulation-to-reality transfer problem is a long-standing challenge in robot learning [15]. In our previous work [16], we employ the DDPG for grasping of an object. The DDPG-based grasping strategy will perform plenty of actions adjustment in training, and has many super parameters need to be adjusted. Avoiding the tedious parameters adjustment, this work considers a grasping action as an end-to-end problem. We pay more attentions on the learning of target not the trajectory for grasping. The Deep Q-learning network (DQN) is then employed to deal with the grasping problem. The input of the model is RGB image, while the output is a robot grasping action. Compared with our previous work [16], there are less parameters need to be adjusted using DQN. And only one decision should be made in a grasping task.

The contributions of this work are summarized as follows:

We learn the robot grasping from a random pile with a Deep Q-network (DQN). To improve the grasping successful rate, each robot grasping action is determined by the result of its current step and the next n steps.

We deal with the "reality gap" from a deep Q-learning policy learned in simulated environments to the real-world by large number of simulation data and small number of real data.

More closely related to our work is that of Zeng et al. [17], which learned synergies between pushing and grasping from experience through Deep Q-learning. They show good grasping performance on the pile of objects by sequence actions that combine the pushing and grasping actions. However, their intrinsic reward of DQN does not explicitly consider whether a push enables future grasps. In contrast, our grasping reward is considered by the current grasp and the next n steps. That is, we also can grasp from a pile of object only with grasping action.

The rest of this paper is arranged as follows: In Sect. 2, some concepts related to RL and DQN are introduced. In Sect. 3, the construction of the DQN-based grasping strategy is presented. In Sect. 4, the performances of the algorithms are given and simulation. In Sect. 5, the method of simulation to real is tested by the experiment on picking a target from a pile of objects.

2 Background

In this section, we introduce several concepts and definitions related to the reinforcement learning and Deep Q-learning network (DQN).

2.1 Reinforcement Learning

Reinforcement learning deals with learning in sequential decision-making problems, where the only feedback consists of a scalar reward signal [18]. In this work, we formulate the robot grasping problem as a Markov Decision Process (MDP) framework like most RL-based grasping learning works.

We denote continuous space Markov Decision Process by the tuple $(S, A, P, r, \gamma, S_{-})$, where S is a set of continuous sates of robot, A is a set of continuous actions of robot, P is a state transition probability function, r is a reward function, γ is a discount factor determining the agent's horizon and S_{-} is a set of initial states. The goal of reinforcement learning is to learn a policy $a_t = \pi(s_t)$ to maximize the expected return from the state s_t, where the return can be denoted by $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ in which T is the horizon that the agent optimizes.

The learning of the policy involves constructing an estimate of the expected return from a given state after taking the action a_t:

$$Q_\pi(s_t, a_t) = E_{r_i, s_i, a_i}\left(\sum_{i=t}^{T} \gamma^{i-t} r_i | s_t, a_t\right) \tag{1}$$

where $Q_\pi(s_t, a_t)$ is the action-value function, which is known as Q-function. A recurse vision of Eq. 1 is given as Eq. 2, which is known as the Bellman equation.

$$Q_\pi(s_t, a_t) = E_{r_t, s_{t+1}, a_i} E[r_t + \gamma E_{a_{t+1}}](Q_\pi(s_{t+1}, a_{t+1})) \tag{2}$$

A more comprehensive introduction of the equation can be found in [18].

2.2 Deep *Q*-learning Network

Deep Q-learning network (DQN) is a value approximation method which combines reinforcement learning with artificial neural network to deal with complex sequential decision-making problems [8].

The update of the Q-network is similar to a typical supervised learning method, and the update at iteration t uses the following loss function:

$$L(\theta^Q) = \frac{1}{N}\sum\nolimits_t |y_t - Q(s_t, a_t|\theta^Q)|_2 \tag{3}$$

where $|\cdot|_2$ denotes 2-norm, θ^Q is the parameters of the function approximators, and y_t can be seen as a 'label':

$$y_t = r_t + \gamma Q(S_{t+1}, \mu(s_{t+1})|\theta^Q) \tag{4}$$

where r_t is immediate reward assigned to each action, γ is the discount factor, s_{t+1} is the next state, $\mu = \underset{a}{\operatorname{argmax}}\, Q(s,t)$.

3 Design of Grasping Strategy with DQN

The problem of grasping an object from a random pile is shown in Fig. 1, where the objects are laid closely side by side, and the robot is required to pick every object. This work proposes the grasping strategy based on DQN, where a CNN is employed to estimate the pose of the objects, and a fully connected network is used to approximate the optimal action-value function.

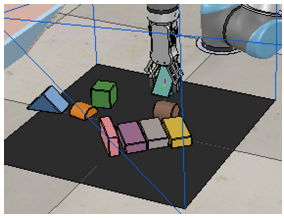
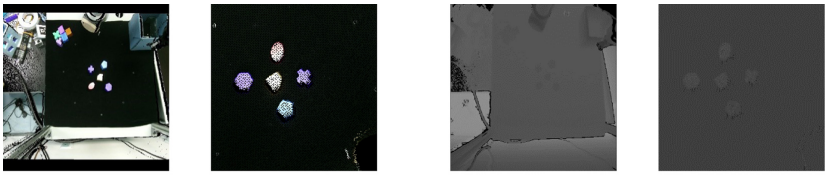


Fig. 1. A target should be picked from a random pile.

3.1 Representations of the Target State

Firstly, a high-dimensional RGB-D image of the target object is used to represent its state. The RGB-D image includes color image and depth image, which are with different views and sizes. As shown in Fig. 2(a), the size of the RGB image is 1920×1080 and the size of depth image is 520×414. An image alignment method is initially employed to match the RGB image with the depth image. The aligned images are shown Fig. 2(b), which make the object position description be consistent in the pixel spatial coordinate frame. It should be noted that the alignment of the images will improve the efficiency of the actions training.



(a) (b)

Fig. 2. The image has been projected onto a 3D point cloud, and orthographically back-project upwards in the gravity direction to construct a height map image representation with both color (RGB) and height-from-bottom (D) channels. (a) shows the RGB image and its height map image, and (b) shows the depth image and its height image.

The rotation angles of the target are separated into 18 orientations, where the interval between two adjacent angles is 20°. The input height map represents the rotation of the input image at 18 different angles, as shown in Fig. 3.

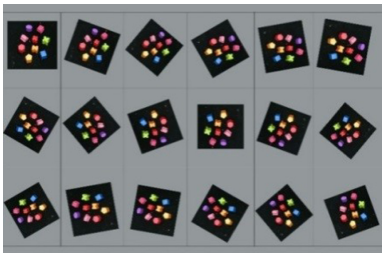


Fig. 3. Input height map represents the rotation of the input image at 18 different angles.

3.2 Representations of the Grasping Action

We represent the robot grasping action by a 4-dimensional vector (x, y, z, r_z), where x, y and z describe the object position in the Cartesian coordinates, r_z is the angle around z axis that is used to control the rotation of the gripper. Note that since the object is grabbed on a horizontal plane, the posture of the object will only change along the z axis, so we fixed the grasping angle r_x and r_y along the x and y axis, and only learned the grasping angle r_z along the z axis.

3.3 Design of the Rewards

Grasping of a target from a pile of objects suffers from the narrow space between objects. To solve the problem, each grasping action is determined by the result of its current step and the next n steps, that is, the current reward is decided by its immediate and the later rewards. It should be noted that if only one object lefts on the table, the immediate reward is determined by the result of the current grasping. Therefore, the immediate reward r_t is expressed as follows:

$$r_t = \sum_{i=0}^{n} \gamma^i r_{t+1} \tag{5}$$

where $\gamma \in [0, 1]$ is a discount factor.

In the training, an RGB-D image is taken as the input state, and the output is a 4-dimensional state-action value function, which describes the target pose of the gripper. The selection of the best action a_t is defined by:

$$a_t = \underset{a}{\operatorname{argmax}}\, Q(s_t, a_t) \tag{6}$$

where the action-value function Q relates to the robot action.

3.4 Design of the Q-networks

This work extends the vanilla deep Q-network by enhancing its feature extraction capability as shown in Fig. 4, where a convolutional neural network is in front of the full connect layers, which is used to extract the features of the input image.

The output $Q(s_t, a_t)$ represents the desired cumulative reward of (x, y, z, r_z) for the action a_t in the current state s_t. Thus, the corresponding action is executed with the maximum Q value. In the network, a resnet-v2 network [19] is employed as feature extraction module, which is pre-trained on ImageNet [20]. The parameters of the shallow network are fixed and only the parameters of the last few layers are tuned in the training.

3.5 Training of the Grasping DQN

Like reference [17], this work employs DQN algorithm to train the grasping action. The grasping features learned by the deep network are visualized as Fig. 4, which represents the optimal grasping point.

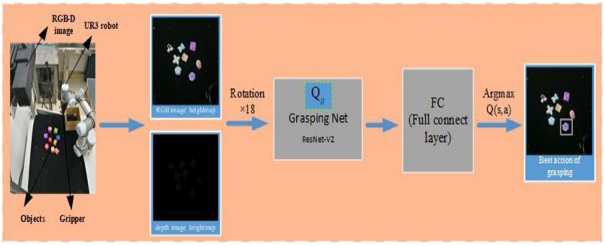


Fig. 4. The Q-networks proposed to train the grasping action.

When grasping from a pile of objects, we hope that the current grasp will be successful and also make the next grasp easily. That is, the current grasp picks up a target object meanwhile change the distribution of objects to make the objects scatter. Thus, the current reward of the grasp is design by the result of the current round and the results of the next episode; hence, each grasping action not only considers whether the current episode is successful, but also considers the results of the following rounds. The current reward (5) is re-written as:

$$R(t) = \sum_{i=0}^{n} \gamma^i r^{t+i} \tag{7}$$

where r is the sparse reward for the grasping action at t , that is, if the grasping action is successful, the reward function r is 1, otherwise it is 0.

$$r = \begin{cases} 1, & if\ successed \\ 0, & otherwise \end{cases} \tag{8}$$

The overview of the whole strategy is given as Algorithm 1.

Algorithm 1. Training of the single grasping action

Initialize parameters of the grasping net Q_g.
for index=1 to N: **do**
  Initialize the simulation environment in random state.
  Capture the RGB-image and Deep-image.
  Process the images and then input them to the network.
  Get the pixel position (p_x, p_y) of each image and obtain the grasping action through coordinate transformation.
  if $t < T_{prop}\%$ where T_{prop} is the exploration probability **then**
    Set a_t=random action as final action.
  else
    Set $a_t = \text{argmax}_a Q_g(s_t, a_t)$ as final action.
    if $a_t == \text{argmax}_a Q_g(s_t, a_t)$ **then**
      Update parameter of Q_g by $\min_{\theta_g}(y - Q_{\theta_g}(s, a))^2$.
    end if
  end if
end for

4 Simulation and Experiment

In this section, simulations and experiments are conduct to illustrate the proposed grasping learning algorithm, and the comparisons between our work and others are also given.

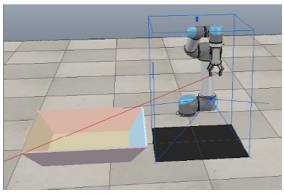


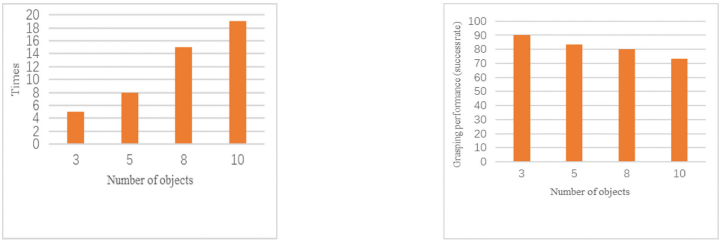
Fig. 5. The simulation of the pushing and grasping actions on V-REP.

4.1 Simulations

We firstly evaluate the proposed algorithm with simulation on V-REP, as shown in Fig. 5. The simulation environment includes a UR5 robot, a RG2 gripper, a Kinect2.0, and a pile of objects with different shapes.

We focus on the performances of the convergence speed, the total number of the grasping, and the success rate of grasping at the same condition. We assume that 10 objects should be picked from the table. The convergence rates of the training is about 75%, and the number of training episodes is about 1000.

Figure 6(a) shows the training times of grasping objects. We test on the grasping of 3, 5, 8 and 10 objects, respectively. If grasping times more than the pre-set threshold, the grasp is still unfinished, then the threshold represents the result.



(a) (b)

Fig. 6. (a) the times for training the grasps; (b) the success rate in grasping of different number of objects.

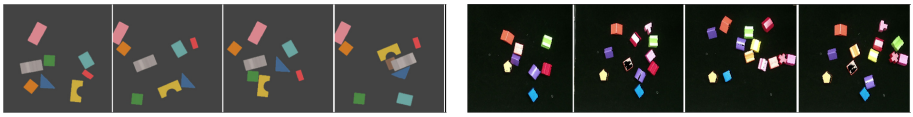
The success rate of grasping is defined the number of objects successfully grasped divided by the total number of objects. We use the average value of 10 grasping as the final success rate of grasping, and test the success rate in grasping of 3, 5, 8, and 10 objects as shown in Fig. 6(b).

We compare the proposed method to the two baseline methods in a simulation where 10 objects are randomly dropped onto a table. Our grasp success is 72%. As reported in reference [17], their grasp success of 30 objects is 67.7%. It is interesting that the success rate of the proposed grasping with long term rewards is similar to that of visual pushing and grasping (VPG).

4.2 Strategy Transferring from Simulation to Real

It is notoriously costly and dangerous to train a real grasp when the deep Q-learning networks have initialized weights values as random. One way to deal with the problem is to leverage the power of simulation to produce the parameters of the deep Q-network. That is, training grasping model on simulation and then transferring them to the real robot. Using domain adaptation methods to cross this "reality gap" requires a large amount of unlabelled real-world data, while domain randomization could waste modelling power [21].

In this work, we deal with the "reality gap" by "large-scale simulation data & small-scale real data". We firstly train the deep Q-network in the simulated environment in which the grasped objects are shown in Fig. 7(a). And then, the deep Q-network is transferred to the real robot for appropriate training in which the grasped objects are shown in Fig. 7(b). The proposed method would avoid the inconsistency between the simulation environment and the real environment, such as the camera installation position, the light information, etc.



(a) (b)

Fig. 7. The objects used in training, where (a) shows the objects used in the simulated environment, and (b) shows the objects used in the real robot.

In this work, it takes about 2000 rounds to train the network in the simulated environment and about 500 rounds in the real environment.

Our model is based on the python deep learning framework and trained on our server, with Intel Xeon E5-2630V3 CPU and Titan XP GPU.

The real robot grasping system for picking of a pile of object is shown in Fig. 8, which includes a Universal Robot 3 (UR3), a Robotiq85 gripper and a Kinect2.0 camera. We develop the grasping software with Python language.

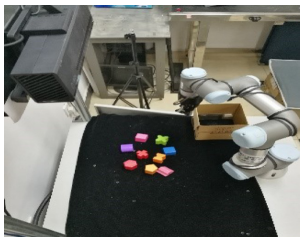


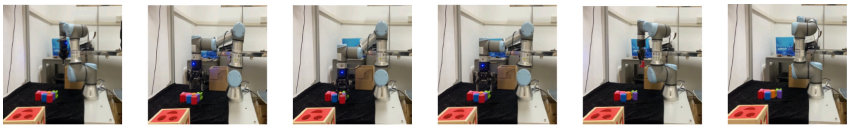
Fig. 8. The real grasping system.

The RGB image and depth image captured by Kinect are with different views and image sizes; hence, we perform an image alignment operation for the RGB image, and map it to the same perspective as the depth image. This operation ensures the consistency of object position description in the pixel spatial coordinate system for RGB image and depth image.

The captured image contains many objects, which will affect the grasping decision of the agent. Thus, we project the image onto a 3D point cloud, and orthographically back-project upwards in the gravity direction to construct a height map image that represents both color (RGB) and height-from-bottom (D) channels. In the real environment, we test the grasping success rate and the generalization ability of the DQN-based grasping model.

In the following, we will describe how to use the algorithm we proposed to control the robot to complete a grasping process in the case of mixed distribution of 8 objects shown in Fig. 9.

Figure 9 shows a complete process of the robot's grasping object. The robot system first initializes, then collects the image from the camera, calculates the position of the object on the plane, and then moves directly to the top of the



(a) (b) (c) (d) (e) (f)

Fig. 9. A pile of objects on the table are picked up by the robot one by one. (a) Robot system initialization. (b) Robot moves directly above the object position. (c) Robot adjusts the gesture of the gripper. (d) Robot is grasping the object. (e) Robot determines if the object is grasped successful. (f) Robot puts the object into the tray.

object. Then the robot began to calculate the position on the z axis and adjust the jaw to grab the object along the z axis. After grasping, the robot determines whether to successfully capture the object through the force of the hand. If the failure is refetched, the captured object is placed in the tray.

5 Conclusion

This paper proposed a robot grasping method based on the Deep Q-network. The full convolutional network is used to extract the input image features, and the full connection layer outputs the action value. Unlike other similar methods, our method determines each robot grasping action through the result of its current step and the next n steps. In the process of grasping, the distribution of objects on the table is constantly changing. Previously obscured objects appear in the robot's field of view. The proposed method still has the ability to generalize to previously unseen objects. To eliminating "reality gap" between simulation and real-world, we use large number of simulation data and small number of real data in the real-world training.

References

1. Goins, A.K., Carpenter, R., Wong, W.K., Balasubramanian, R.: Evaluating the efficacy of grasp metrics for utilization in a gaussian process-based grasp predictor. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3353–3360 (2014). https://doi.org/10.1109/IROS.2014.6943029
2. Boularias, A., Kroemer, O., Peters, J.: Learning robot grasping from 3-d images with markov random fields. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1548–1553 (2011). https://doi.org/10.1109/IROS.2011.6094888
3. Balaguer, B., Carpin, S.: Learning end-effector orientations for novel object grasping tasks. In: 2010 10th IEEE-RAS International Conference on Humanoid Robots, pp. 302–307 (2010). https://doi.org/10.1109/ICHR.2010.5686826
4. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. Int. J. Rob. Res. **27**(2), 157–173 (2008)
5. Bohg, J., Morales, A., Asfour, T., Kragic, D.: Data-driven grasp synthesis-a survey. IEEE Trans. Rob. **30**(2), 289–309 (2013)

6. Pinto, L., Gupta, A.: Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 3406–3413. IEEE (2016)
7. Mahler, J., et al.: Learning ambidextrous robot grasping policies. Sci. Rob. **4**(26) (2019)
8. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
9. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
10. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396. IEEE (2017)
11. Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., Levine, S.: Visual reinforcement learning with imagined goals. arXiv preprint arXiv:1807.04742 (2018)
12. Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., Abbeel, P.: Asymmetric actor critic for image-based robot learning. arXiv preprint arXiv:1710.06542 (2017)
13. Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., Riedmiller, M.: Data-efficient deep reinforcement learning for dexterous manipulation. arXiv preprint arXiv:1704.03073 (2017)
14. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research **37**(4–5), 421–436 (2018)
15. Sadeghi, F., Toshev, A., Jang, E., Levine, S.: Sim2real view invariant visual servoing by recurrent control. arXiv preprint arXiv:1712.07642 (2017)
16. Chen, B., Su, J.: Addressing reward engineering for deep reinforcement learning on multi-stage task. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. CCIS, vol. 1143, pp. 309–317. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36802-9_33
17. Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., Funkhouser, T.: Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4238–4245. IEEE (2018)
18. Wiering, M., Van Otterlo, M.: Reinforcement learning. Adapt. Learn. Optim. **12**(3) (2012)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
20. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
21. James, S., et al.: Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12627–12637 (2019)