University of Birmingham

School of Engineering
Department of Mechanical Engineering

Individual Engineering Project

FINAL REPORT

BEng

Surname	Long
First Name(s)	Yunbo
ID number	2226397
Supervisor's Name	Dr. Yongjing Wang
Project Title	Deep reinforcement learning for virtual industrial disassembly with visual and haptic perception

Abstract

Industrial disassembly tasks make manual approaches to designing robotic control models complicated due to their complexity and the richness of contact operations. While there is much current research on autonomous learning methods for industrial robots, little research combines perceptual methods such as vision and haptics for industrial disassembly tasks. This paper investigates how a robot learns autonomously and completes an industrial disassembly task by combining deep reinforcement learning algorithms with multidimensional perceptual capabilities such as vision and haptics. By analyzing the performance of the autonomous learning system on the disassembly task of a door chain model and comparing the results of the robot's autonomous learning with different reward functions, the study not only demonstrates the flexibility and speed of the robot's autonomous learning model for the industrial disassembly task but also finds that the sparsity of the reward function has an important impact on the stability of the autonomous learning model. The proposed robot autonomous learning model can help provide a solution for industrial contact-rich tasks that can be quickly validated.

1. Introduction

With machine learning techniques proving the possibility of autonomous learning by agents in areas such as manufacturing [1], more and more research is being conducted using machine learning techniques, particularly reinforcement learning algorithms, to improve assembly techniques for industrial robots [2], [3], [4]. However, cases of their application to disassembly tasks are still rare [5], [6]. Unlike assembly, industrial disassembly is more challenging as robots face uncertainties associated with different product models and environmental changes [7], [8] and need to make complex trajectories in the face of unknown physical structures[9]. For technical feasibility and economic cost, current industrial disassembly tasks are often carried out manually by humans [10] or through human-robot collaboration [11] due to the high degree of flexibility required by the strategy. In addition, the automated control strategies of conventional robots are often costly, requiring not only the design of precise robot control schemes for complex systems but also frequent changes to the control strategy under conditions such as changes in the relative position of the workpiece [4], [12].

It is, therefore, essential to find a method of industrial dismantling strategy that is flexible and stable in operation. However, in industrial tasks, where the robot's environment is three-dimensional and the motion of the robot arm is multi-jointed, traditional reinforcement learning methods are not applicable in the face of such high-dimensional and complex variables. Specifically, in contact-rich tasks such as industrial disassembly, the two critical perceptual dimensions are vision and haptics [1]: the visual perception of the 3D coordinates of an object for localization requires images containing high-dimensional pixel points. In contrast, the haptic information guiding the precise disassembly of the robot arm requires multi-dimensional spatial force analysis. Deep neural networks need to be introduced to explain to these realistic and complex models so that reinforcement learning algorithms can be applied to real disassembly tasks [13].

In this paper, in order to quickly validate and explore the application of deep reinforcement learning algorithms to disassembly tasks, a realistic physical simulation environment was constructed to meet the need for multidimensional perceptual capabilities, and the Proximal Policy Optimization (PPO) algorithm was used to train a robotic arm to complete the disassembly task of a door chain lock. The impact of the reward function on the robot's autonomous learning was then explored by comparing the stability of different reward functions on the robot's performance during the disassembly process. This study demonstrates the feasibility of autonomous learning robots for industrial disassembly tasks and provides the basis for further research, such as future sim-to-real experiments.

2. Methods

2.1. Theoretical framework

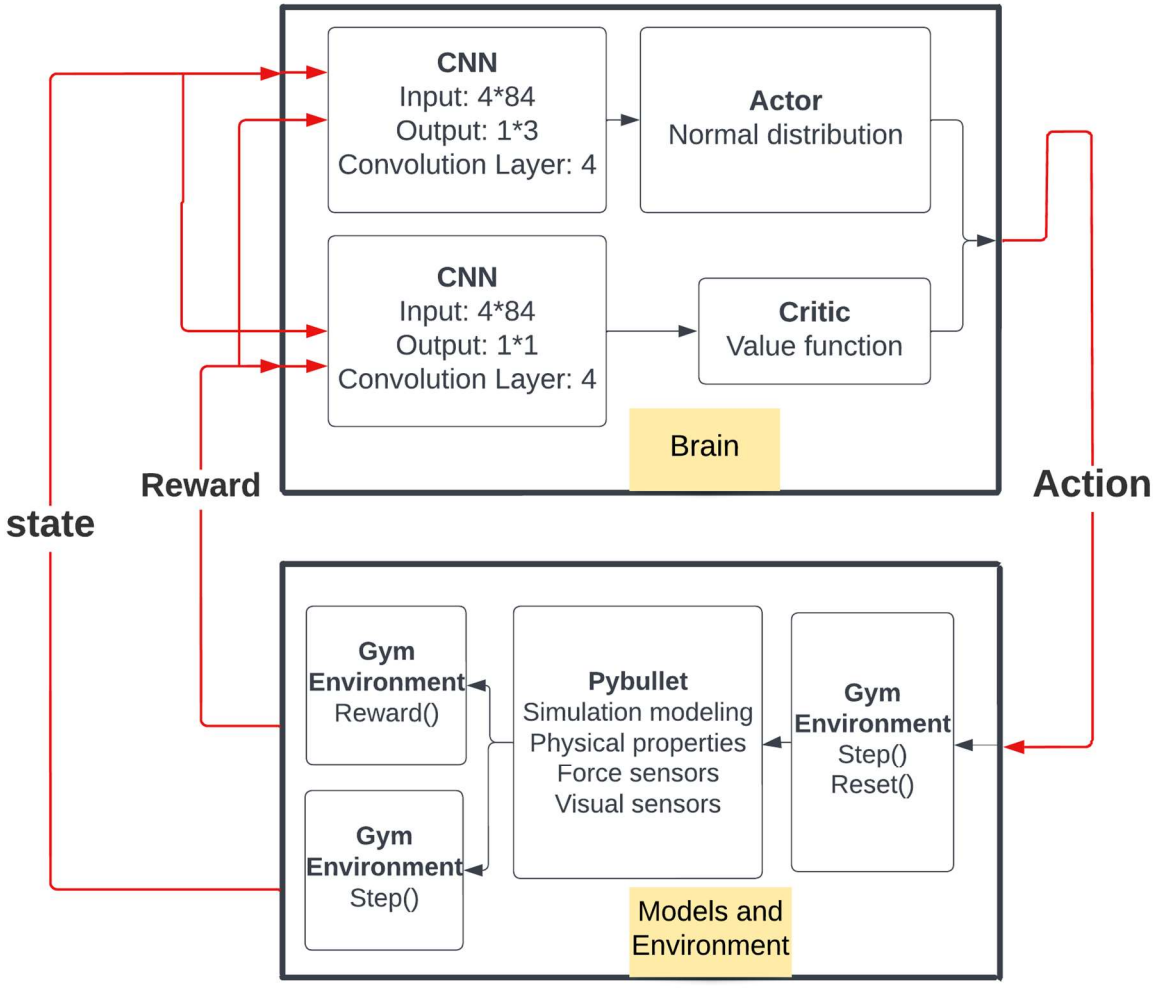


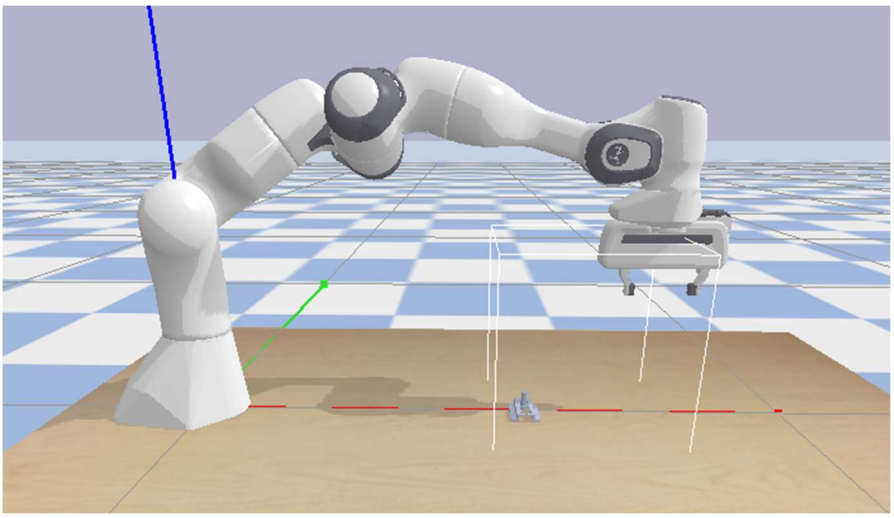
Figure. 1: Theoretical framework of autonomous learning systems for robots

The diagram shows the core theoretical framework for the autonomous robot learning system in this paper. In summary, accurate model physical properties and the inclusion of multi-dimensional sensors in the model and environment construction accurately reproduce the

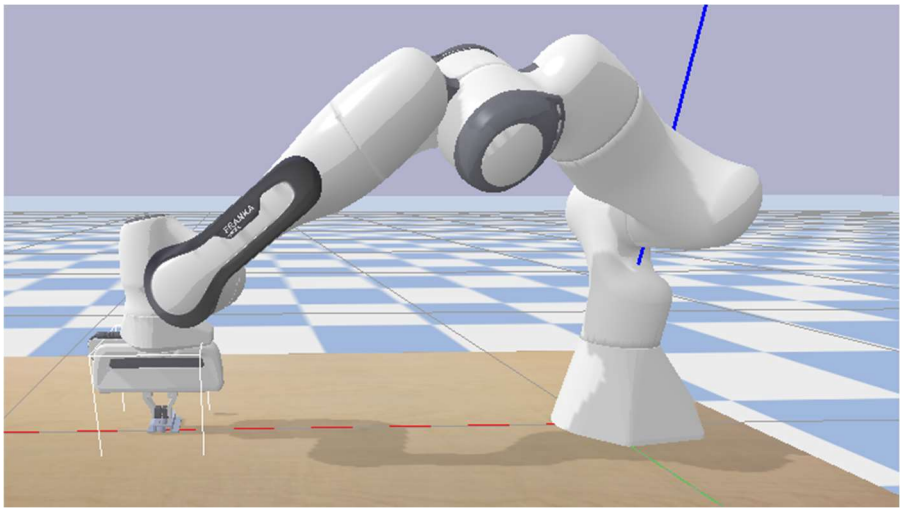
interaction of the robot arm with the industrial disassembly task scenario. Furthermore, in the brain of the agent, convolutional neural networks (CNN) are used to perceive the environment and combined with PPO algorithms to build the intelligent body's brain. During the model's training, the intelligent body learns the industrial dismantling task autonomously by taking different action instructions and receiving different reward values and state update parameters. These aspects are described in detail in the following sections.

2.2. Physical environment modelling

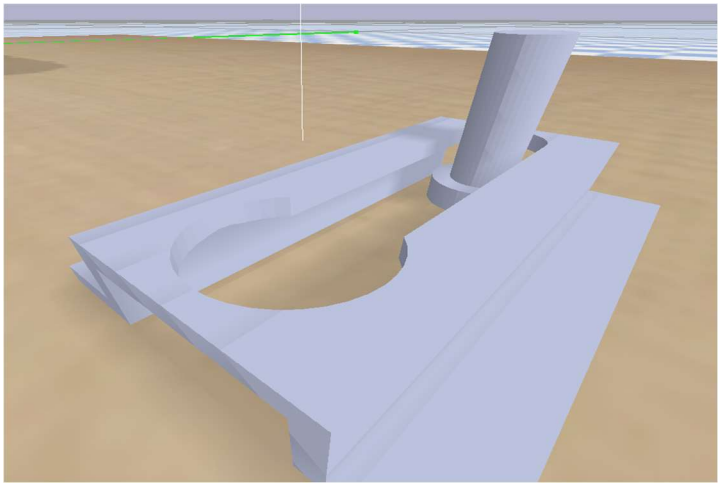
There was a laboratory table on which a robotic arm with a fixed base and a slot where the base was also fixed were placed in the gravity space. Also, a non-fixed bolt was placed at a designated location, and several white lines were drawn around the bolt to limit the area where the chain lock device would appear and the area where the gripper could move. It can be noticed that the white area in the bolt gripping experiment is much larger than in the bolt removal experiment. This is because the large area of the former allows the autonomous learning system to be applied to a broader range of bolt gripping environments. In contrast, the small area of the latter allows the robot to learn the bolt removal process more quickly. The specific experimental simulation environment and chain lock model are shown below.



(a) Simulation environment of the bolt grasping experiment



(b) Simulation environment of the bolt removal experiment



(c) Simulation environment of the disassembled device

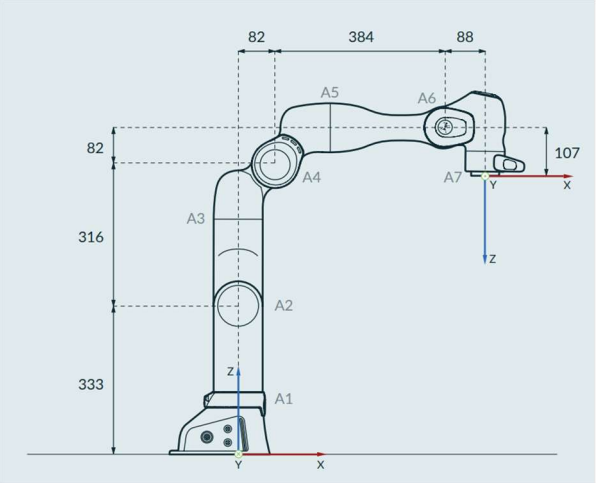
Figure. 2: Simulation environment for experiments on robotic door chain removal tasks

2.2.1. Selection of the robot and dismantled device

The model used in the experiments is the Franka Emika (Panda), whose sensitive and flexible arm has seven degrees of freedom and torque sensors in each joint, so it was specifically designed to engage in experiments such as torque control. The robotic arm used in this experiment simulates a realistic robotic arm disassembly process, so the robotic arm model must feature highly accurate contact recognition, interpretation, and response, and the Panda robot fulfills this need. It is noticeable that this experiment used the self-contained robot URDF information in Pybullet to load the CAD model of the Panda robot [14]. The dimensions and specific parameters of its CAD model are shown in figure 3.

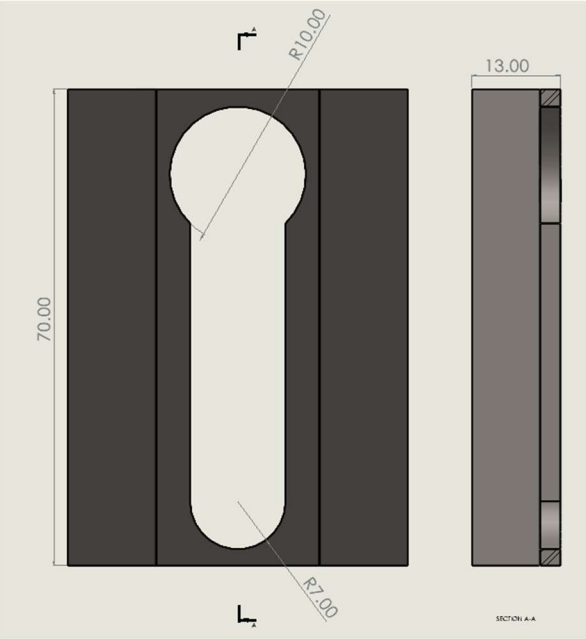
The recessed shape is very similar to the door chain lock model. A tube with a larger diameter at one end of the bolt than the slot is placed on the inside, making it impossible for

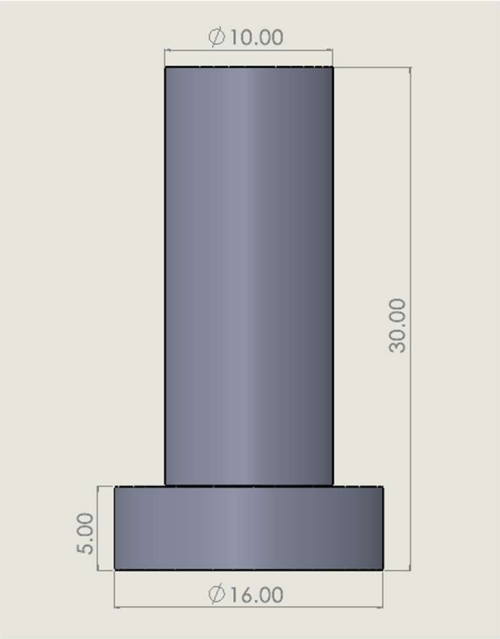
the bolt to be pulled out directly. However, at the other end of the slot, there is a circular opening larger than the head of the bolt, which is the only exit for the bolt to be removed. The dimensions of the bolt and slot are essentially the same as the physical reality of the dimensions in the laboratory. The exact CAD model and parameters are also shown in figure 3.



Degrees of freedom	7
Payload	3 kg
Maximum reach	855 mm
Force/Torque sensing	link-side torque sensor in all 7 axes
Joint position limits	A1, A3: -166/166 deg
	A2: -105/105 deg
	A4: -176/-7 deg
	A5: -165/165 deg
	A6: 25/265 deg
	A7: -175/175 deg
Mounting flange	DIN ISO 9409-1-A50
Installation position	upright
Weight	~ 17.8 kg

(a) Dimensions of the Panda robot. *(b) Parameters of the Panda robot.* [15]





(c) Dimensions of the slot *(d) Parameters of the bolt*

Figure. 3: Basic parameters of the physical model of the simulation

2.2.2. Key physical parameters settings

For most physics simulation software like Simscape in MATLAB, to reduce computational effort and optimize unimportant details, the system defaults to using a convex hull instead of a concave mesh when building a 3D physics model [16]. This is not feasible for accurate physical simulations. In contrast, the Pybullet physics simulation [17] is able to preserve any initial shape of the robot arm through XML statements in the URDF file.

Also, the simulation experiment differs from other robotics algorithm application projects in that it looks more at how to restore the realism of industrial rich contact tasks and therefore emphasizes the importance of physical properties such as object mass, material stiffness, and surface friction in the physical simulation, which are important for realistic simulations of industrial dismantling tasks. The essential physical properties of the experimentally relevant models are shown below.

Table I

Physical properties of the models

Physical properties (Gravity=-10m/s^2)				
	Table	Bolt	Fixed notch	Panda robot
Material	Steel	Copper	Steel	Steel
Contact coefficients (mu - Friction coefficient, kp - Stiffness coefficient , kd - Dampening coefficient)	mu = 0.4 kp = 1000 kd = 1.0	mu = 0.5 kp = 600 kd = 1.0	Mu = 0.4 kp = 1000 kd = 1.0	/
Significant parameters of Panda (for each joint)				
	K position	K velocity	Soft lower limit	Soft upper limit
Panda joints (Safety controller)	100.0	40.0	-2.8973	2.8973

	Stiffness	Damping	Spinning friction	Lateral friction
Friction (both left and right fingers of Pandas)	30000.0	1000.0	0.1	0.5

2.3. Virtual experiments design

In order to study in detail the use of deep reinforcement learning algorithms by robots for industrial dismantling tasks, a case study was set up for this experiment, a model of a robot arm autonomously learning how to dismantle a door chain lock. Compared to physical experiments using physical objects, the virtual simulation experiment avoids damage to experimental equipment and human safety due to experimental anomalies. It enables rapid iteration and validation of the prototype's algorithms. In addition, when designing industrial robot tasks, previous research has typically mounted the robot arm at precisely the specified location around the dismantled equipment and then controlled the positional movement of the end of the robot arm by solving for the robot's motion trajectory. This experimental approach requires precise distance measurement between the disassembled equipment and the robot base and estimation of the most appropriate motion trajectory. However, its fixed thinking is very susceptible to environmental changes.

Therefore, this experiment studies an autonomous learning method that allows the robot to adopt an appropriate strategy when faced with a model of a door chain lock that appears in an arbitrary position. Unlike most industrial simulation tasks, this experiment uses reinforcement learning algorithms and introduces visual and tactile perception to facilitate the autonomous learning process. In addition, a control group with a sparse reward function was set up to investigate the effect of reward function sparsity on the stability of the robot learning model. In order to reduce the training time of the model and to set the reward values for the different phases, the whole door chain lock removal experiment was divided into two phases, including the bolt grasping and bolt removal phases.

2.3.1. The bolt grasping experiment

The first experiment uses a deep reinforcement learning algorithm to train the robotic arm at a fixed initial position to master how to quickly grasp a bolt in a door chain locking device,

where the bolt and its device are randomly present in a flat area enclosed by a white line and a tabletop during each round of training. The results were evaluated through experimental simulation animations, and cumulative performance value curves were returned for each episode. In addition, two sets of experiments with different reward functions were set up, and the two experimental models were trained in five separate simulations. The points of the resulting curves were analyzed for a variance to assess the effect of the reward functions on the stability of the robot learning models.

2.3.2. The bolt removal experiment

Following the first experiment, this experiment trained the robot arm to remove the bolt from the other end of the door chain lock model after grasping the bolt while maintaining the grasping position. In particular, the initial positions of the door chain lock model and the robot arm are fixed. This experiment focuses on how parameters such as friction and positive pressure are introduced into the reinforcement learning reward function to help the robot achieve this rich contact task. Finally, the feasibility of the model is illustrated by comparing the simulation results curves.

2.4. Perceptual system building

The implementation of visual perception and tactile perception has led to the robot being able to reinforce its learning process at the macro level of image perception and the subtle level of force analysis, and these two capabilities are complementary.

2.4.1. Neural networks for visual processing

As the basis of neural networks, multilayer perceptrons (MLPs), also known as artificial neural networks (ANNs), are widely used in machine learning for data classification tasks [18]. However, MLPs can only accept vector input when processing data. In this simulation experiment, a convolutional neural network (CNN) that accepts a matrix as input was therefore introduced in order to be able to read the spatial information between pixels. This is supplemented by a pooling layer and a fully connected layer to achieve a transformation from the input of the image to the mathematical distribution of the desired output values.

In this experiment, the MLP programming model in the spin algorithm library was modified to a CNN model. However, the default image size was set to 960*960, which is huge for neural network training. It can be known that the most suitable size for the input image of a neural network is 84*84, and the number of its channels is reduced by converting RGB to

greyscale again [19]. Again, since previous studies have proved that image data augmentation methods such as Crop are very helpful for deep neural network training, this experiment sets the initial input image size to around 100*100 and then further optimizes it to 84*84 by data augmentation methods such as Crop to meet the needs of deep neural network training [20]. The deep neural network consists of 32 superimposed images convolved in four convolutional layers, where the initial input image size is 84. In addition to this, all convolutional layers use the same values for important parameters including a kernel size of 3, a stride of 2 and a padding of 1. After the convolutional layers, the network enters the fully connected layers. It is worth noting that each convolutional layer and one fully connected layer of the neural network is followed by a non-linear activation function, ReLU function, to enhance the fitting ability of the network. The size of the feature map output during convolution is formulated as follows.

$$Output\ size = \frac{(Image\ size - Knernel\ size + 2 \times Padding)}{Stride} + 1 \qquad (1)$$

The values calculated for the essential parameters of this convolutional neural network are therefore shown in the table 2. The critical parameters for each layer were calculated are shown in figure 4.0.

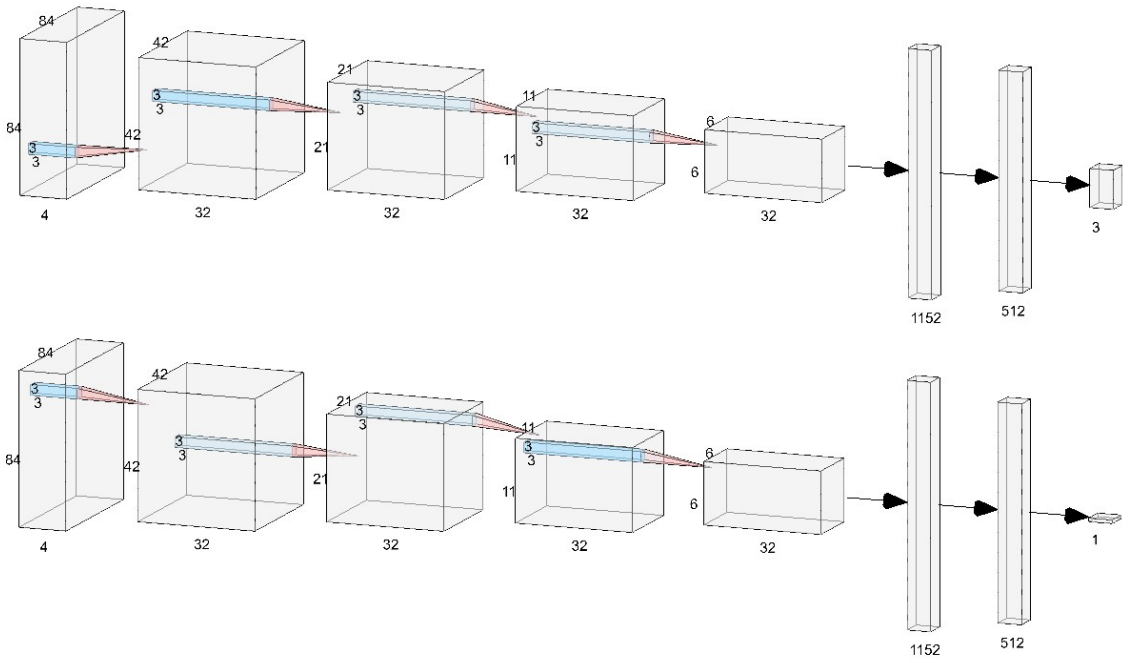


Figure. 4: The two convolutional neural networks

2.4.2. Principle of haptic feedback

In this experiment, the Pybullet simulation engine returned the contact points calculated at the last call to stop simulation via the getContactPoints API. The return values include the normal force and the amount of friction in the x-axis and y-axis directions. Variables were set up for this experiment to detect the normal force between the jaws of the panda robot and the bolt and the frictional force between the bolt and the inner wall of the slot, respectively. These parameters will influence the design of the reward function later on.

In particular, in real-life physics experiments, the robot arm is equipped with force sensors at the end of the arm to detect the real force data on that point and to further calculate the force between the arm and the bolt and the groove through mathematical coordinate conversion equations. In the simulation, these interaction forces are obtained directly from the system. Although this model will be inconvenient for future sim-to-real experiments, it will allow for quick verification of the algorithm model in this experiment.

2.5. Reinforcement learning algorithm modelling

Reinforcement learning algorithms are widely used in the autonomous learning process of agents, which contains the essential parameters that explain the core ideas of their learning. Furthermore, in this experiment, because of the high latitude complexity of the robot's perception and its motor system, these experiments were finally accomplished by introducing deep reinforcement learning algorithms.

2.5.1. Problem formulation in reinforcement learning

The core of this experiment revolves around the reinforcement learning algorithm, and the basic parameters involved are encapsulated in the Gym environment [14]. Therefore, it is essential to set reasonable reinforcement learning parameters for the Panda robot and the gate chain disassembly model. In particular, experiments should also consider the reasonableness of the reward values when setting up different reward functions.

2.5.1.1. States and Observations

In this experiment, the state S is a complete description of the robot's state. Observation O is the observation of an image in space, using a real-valued vector, matrix, or higher-order tensor to represent the state and observation. Specifically, the robot's state can be represented by its joint angle and velocity, while the visual observation is represented using a pixel-valued RGB matrix.

2.5.1.2. Action Spaces

The space in which an agent controls the movement of a robot in the physical world is continuous. In a continuous space, actions are vectors of actual values. The motion space in the simulation is therefore represented by creating continuous three-dimensional variables in the space of the end of the robot arm. To change the position of the robot arm end coordinates, it needs to be solved by robot inverse kinematics [21]. For the robot to pick up the bolt correctly, the number of joints available to the robot for the inverse kinematics calculation is set in the simulation to 7. After that, the spatial distance between the bolt and the center of mass of the robot's gripping jaws is calculated at each step, and similarly, the distance between the bolt and the specified exit position of the slot is also calculated during the robot disassembly.

2.5.1.3. Stochastic Policies

A policy is a rule that an agent uses to decide what action to take. In this case, it is stochastic, in which case it is usually denoted as. Moreover, the two most common stochastic policies in deep RL are the classification policy and the diagonal Gaussian policy. In general, classification strategies can be used for discrete action spaces, while diagonal Gaussian strategies can be used for continuous action spaces, and the latter is used in this experiment [22].

2.5.1.4. Reward and Return

The reward function R is crucial in reinforcement learning. It depends on the current state of the world, the action that has just been taken, and the next state of the world. However, the reward function can only be set artificially, so the problem of too sparse a reward function can arise. Furthermore, since the agent aims to maximize some notion of cumulative reward on the trajectory, the agent sometimes persists to a local minimum and fails to complete the task [23].

For the bolt grasping experiments, in addition to a sparse reward control group with a reward of 0 in the last condition, the experimental group was set up with an inverse proportional function related to distance and a positive proportional function related to the current cumulative number of steps, where the distance function using the inverse proportional approach was able to produce a greater change in reward when reducing the same distance than the positive proportional function, thus stimulating the tendency of the agent to reduce the distance. At the same time, the agent learns how to take the minimum number of steps to achieve the goal. In the bolt removal experiment, in the last condition, the distance factor, the bolt-slot friction factor, and the bolt-jaw pressure factor are all taken into account in the reward

function so that the bolt remains in contact with the jaws of the robot arm as it moves towards the exit while minimizing the friction between the bolt and the slot wall.

Table II

Simulation of the reward function for Experiment 1

Reward function setting for robotic arm picking simulation experiments		
Serial number Condition	1	2
Manipulator gripping jaws out of white space area	Reward=-0.1 Round End	Reward=-0.1 Round End
Maximum turn length of arm movement exceeds 700	Reward=-0.1 Round End	Reward=-0.1 Round End
Distance between robot arm gripper and bolt less than 0.01m	Reward=1 Task complete	Reward=1 Task complete
In cases other than the above	Reward=0 Task continues	$Reward = \frac{0.1}{Current\ distance1} - \frac{Current\ steps}{50}$ where the current distance 1 here represents the distance between the jaws of the robot arm and the bolt. Task continues

Table III

Simulation of the reward function for Experiment 2

Reward function setting for robotic arm disassembly simulation experiments		
Serial number Condition	1	2
Manipulator gripping jaws out of white space area	Reward=-0.1 Round End	Reward=-0.1 Round End
Maximum turn length of arm movement exceeds 700	Reward=-0.1 Round End	Reward=-0.1 Round End
Distance between the target point and bolt less than 0.01m	Reward=1 Task complete	Reward=1 Task complete
Positive pressure between arm jaws and bolt equals 0N 5 times in a row	Reward=-0.1 Round End	Reward=-0.1 Round End
The force between arm jaws and bolt is greater than 5N	Reward=-0.1 Round End	Reward=-0.1 Round End
Combined frictional force between bolt and inner wall of slot greater than 5N	Reward=-0.1 Round End	Reward=-0.1 Round End
In cases other than the above	Reward=0 Task continues	$Reward = W_1 \times \frac{Initial\ distance}{Current\ distance} - W_2 \times \frac{F_f}{F_{fmax}} + W_3 \times \frac{F_N}{F_{Nmax}}$ $where\ W_1 = 0.4,\ W_2 = 0.3,\ W_3 = 0.3.$ And the current distance here represents the distance between the exit point and the bolt

		Task continues

2.5.2. The Introduction of Deep Reinforcement Learning

2.5.2.1. Algorithm selection

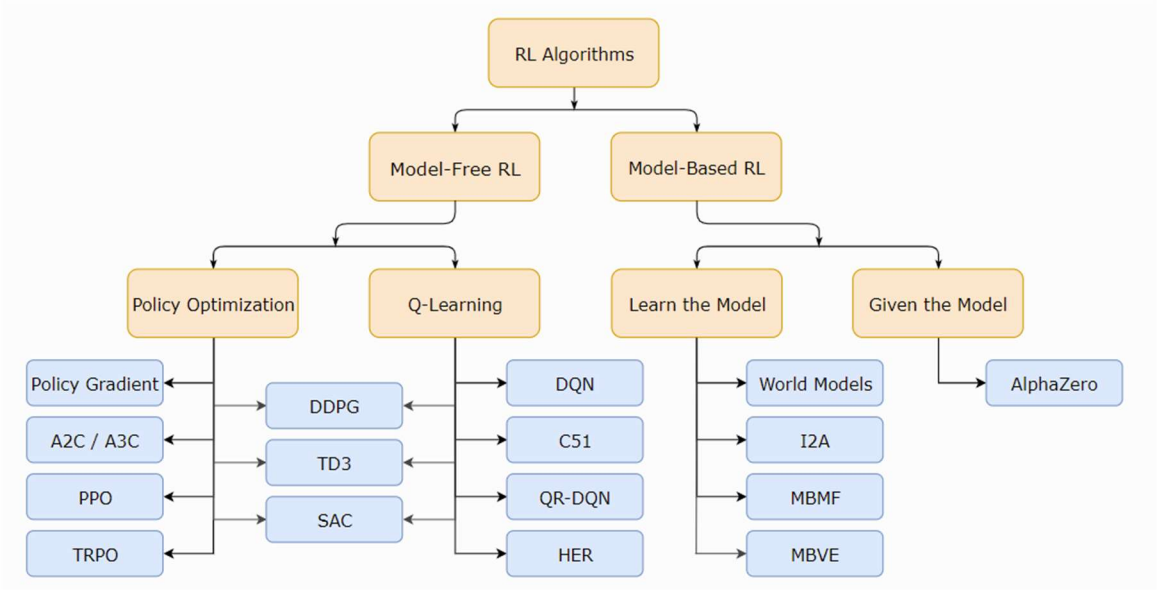


Figure. 5: Classification methods for reinforcement learning. [17]

After establishing the physical simulation environment, the perception system under the neural network, and the underlying reinforcement learning environment, it is crucial to introduce deep neural networks and select deep reinforcement learning algorithms. As shown in figure 5, deep reinforcement learning algorithms can also be divided into model-based and model-free approaches, depending on whether they are model-based [25]. Although in the model-based approach, the experience gained by the robot can be utilized multiple times and the data samples are highly utilized, the generalization of the acquired experience is poor, as the modeling of the environment is biased and will fail once new changes occur. The model-free approach interacts directly with the environment, has no modeling bias, and is highly generalizable [26]. In order to improve the generalization capability of the industrial robot during disassembly, the mod-free approach was adopted in order to improve the generalization capability of the industrial robot during disassembly, as actual industrial tasks are associated with uncertainty and error.

Depending on how the policy is updated, deep reinforcement learning algorithms can also be classified as value-based and policy-based approaches [27]. The most significant disadvantage of the value-based approach is that it is difficult to extend to the continuous action space and suffers from the problem of high bias, where the error between the estimated value function and the actual value function is challenging to eliminate. Unlike value-based policy methods such as DQN, policy-based policy methods do not require estimating the value function. Instead, use neural networks to fit the policy function directly and update the policy parameters through training to generate the best policy. The most representative of these methods is the policy gradient. However, strategy gradient methods require expectation operations when calculating the gradient. High variance is introduced when calculating the expectation for the cumulative rewards of log probabilities for different trajectories, resulting in high noise in the gradient calculation [28]. As a result, the Actor-Critic algorithm has emerged as an algorithm that combines value-based and policy-based approaches, addressing to some extent their respective shortcomings. In particular, based on the Actor-Critic algorithm, the PPO algorithm uses the log probability to calculate the policy gradient, making the calculation more straightforward [29].

2.5.2.2. Theoretical implementation of the algorithm

The PPO algorithm consists of two main parts, the Actor and the Critic. Theoretically, the Actor part is an upgrade of the Policy Gradient algorithm. Considering that as an on-policy algorithm, the PI and Theta of the data sampled can only be used once to update the parameters, so the sample efficiency is very low. The actor part of the PPO uses the importance sampling method, which can estimate the mathematical distribution of the objective function using the distribution of a set of weighted samples [30], as follows.

$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}] \qquad (1)$$

where the two sides of the equation are the two expectations based on the p and q distributions respectively. In the old state where the strategy is represented as π and the parameter θ is updated, the sampling method is changed from one sampling to multiple sampling and the parameter is updated multiple times. In particular, PPO is still an on-policy algorithm, because the distinction between on-policy and off-policy depends on whether the algorithm follows a policy function pi when selecting the current behaviour and when updating the next policy function Q. If the policy function PI is followed in both steps then it is an on-policy algorithm like Sarsa. The policy of the PPO algorithm is always based on the policy function PI, so it is still an on-policy algorithm. Despite this, the importance sampling method used in the PPO only guarantees that the means of the equation transformations are the same, but the variances are different. The variance of the strategy gradient can be calculated as

$$Var_{x\sim p}[f(x)] = E_{x\sim p}[f(x)^2] - (E_{x\sim p}[f(x)])^2 \qquad (2)$$

And the variance of PPO algorithm is

$$Var_{x\sim p}\left[f(x)\frac{p(x)}{q(x)}\right] = E_{x\sim p}\left[f(x)^2\frac{p(x)}{q(x)}\right] - \left(E_{x\sim p}[f(x)]\right)^2 \quad (3)$$

The two sides are identical only when p(x) = q(x). And because those two are completely different functions, the Clip function is used to keep p(x) and q(x) under control so that the objective function of the PPO becomes

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\ clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right) \quad (4)$$

Where π_θ is a stochastic policy, and A_t is an estimator of the advantage function. The values in objective function L are discussed separately for the two cases A>0 and A<0. The final red line in the graph is the objective function value is shown in figure 6.

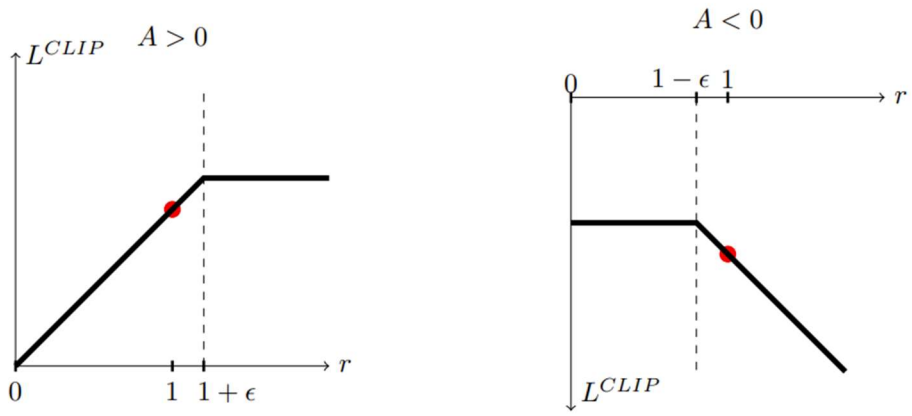


Figure. 6: Image of the objective function of the PPO clip algorithm. [22]

Thus, the whole algorithm of the PPO is finally obtained, which means that the reward is first calculated by sampling in the environment, then once the function is calculated, after which the PPO-clip updates the function, and finally, the value parameter is updated. The policy and the value function are determined by the results obtained from the two previous deep neural networks. The specific algorithm is shown in the figure below. For the meaning of other parameters, please refer to [29].

Algorithm 1 Standard Proximal Policy Optimization Algorithms (PPO-Clip)

Input : initial policy parameters θ_0, *initial value function parameters* ϕ_0

for $k = 0, 1, 2, ...$ **do**

Collect set of trajectories $D_k = \{\tau_i\}$ *by running policy* $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards $-$ *to* $-$ *go* $\hat{R}_t$.

Compute advantage estimates, A_t *(using any method of advantage estimation) based on the current value function* V_{ϕ_k}.

Update the policy by maximizing the PPO $-$ *Clip objective :*

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{n=1}^{N} min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t))),$$

typically via stochastic gradient ascent with Adam.

Fit value function by regression on mean $-$ *square error :*

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2,$$

typically via some gradient descent algorithm.

end for

Figure. 7: PPO-Clip algorithm. Adapted from [29]

2.5.2.3. Essential parameters and the training process of the algorithm

The Spinning-up algorithm library provides the base source code for PPO. In Spinning up, there are core files and PPO algorithm files that are further modified to meet the requirements of the simulation experiments. The experimental algorithms were trained in the CPU as MPI parallel computing mainly uses the CPU. The size and number of layers of the hidden layers and the gamma parameters were sampled from the default parameters of Spinning-up. Thus, PPO can distribute all steps equally to each processor and can use MPI for ambient parallel computation, which speeds up model training. As a rule of thumb, the training environment is divided among the four CPU cores for parallel computation (tests have shown that out of a total of 16 cores on the computer used for the simulation experiments, only up to four can be enabled for smooth running training). The data collected by the model is stored in the spinning up algorithm library. In order to accurately represent the training results for a significant epoch, we use the Matplotlib library to plot the data results for all significant epochs of training. In addition to this, the hyperparameters used in the PPO algorithm are shown in the table below.

Table IV

Hyperparameters used in the the Proximal Policy Optimization (PPO) algorithm

Hyperparameter	Value
Epoch	50
Steps per Epoch	4000
Discount Factor Gamma	0.99
Clipping	0.2
KL Target	0.01
GAE Parameter Lambda	0.97
Learning Rate for Policy Optimizer	0.0003
Learning Rate for Value Function Optimizer	0.001

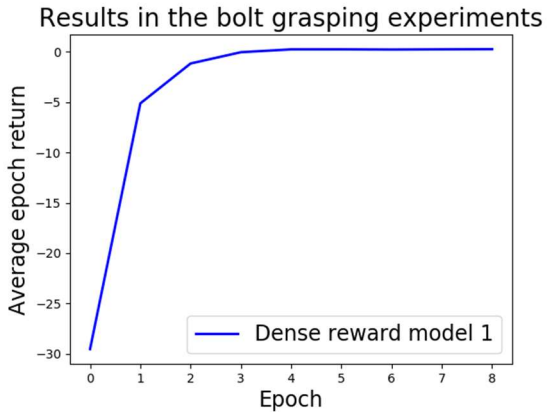
3. Results

The experimental results are divided into the results of the model animation simulation and the numerical results of the model training, both of which are distributed to reveal the results of the experiments from the point of view of qualitative and quantitative analysis.

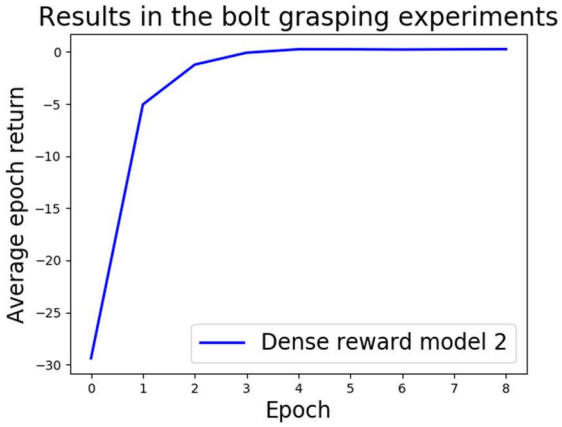
3.1. Results of the bolt grasping experiment

In the bolt gripping experiments, the animated simulation results show that the robot arm in the specified position can quickly locate the bolt in the door chain model and approach it quickly from the preferred path. The training results of the algorithm are shown in Figures 8 and 9, which include an experimental group with dense rewards and a control group with sparse reward functions, both of which include six sets of data plots: five separate data plots and one set of data comparison plots. All data show a trend of convergence.

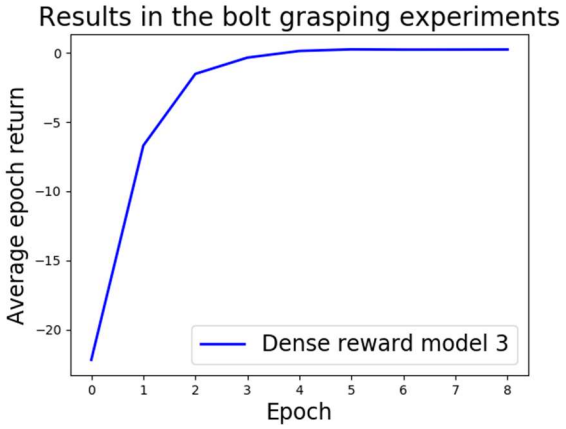
To further explore the influence of the density of the reward function on the stability of the model algorithm, the variance was calculated for the five data distributions of the experimental and control groups, respectively, and the variance curves and the specific numerical magnitude of the variance obtained are shown in Figure 9 and Table 6, respectively.



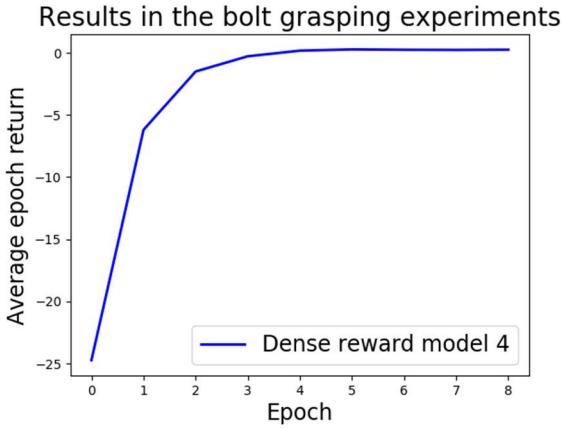
(a) Model 1 with the dense reward



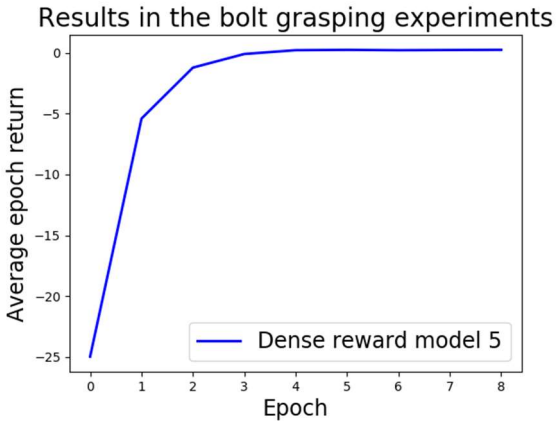
(b) Model 2 with the dense reward



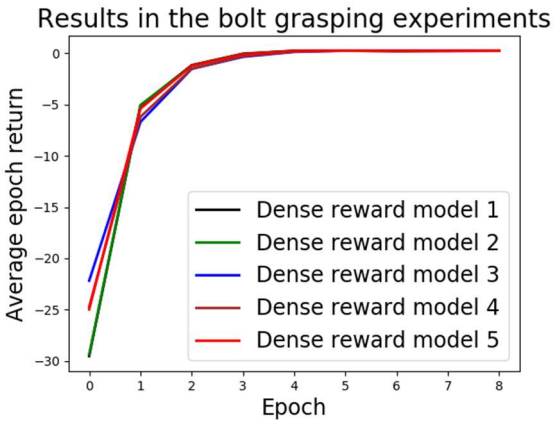
(c) Model 3 with the dense reward



(d) Model 4 with the dense reward

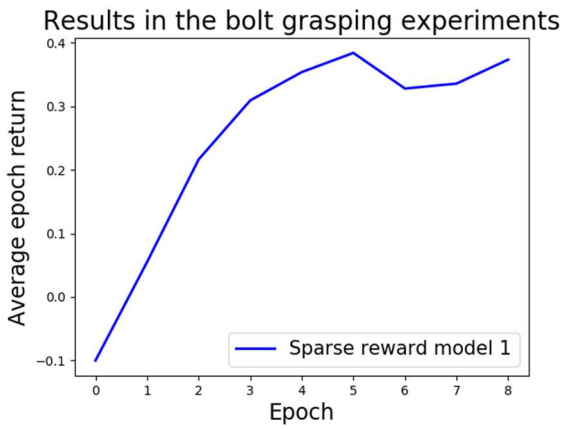


(e) Model 5 with the dense reward

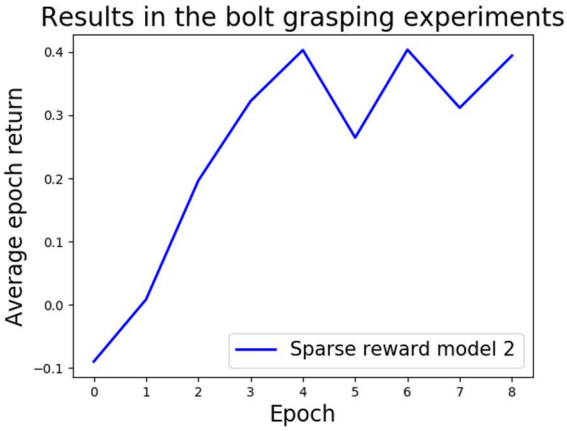


(f) Comparison of 5 models in the dense reward

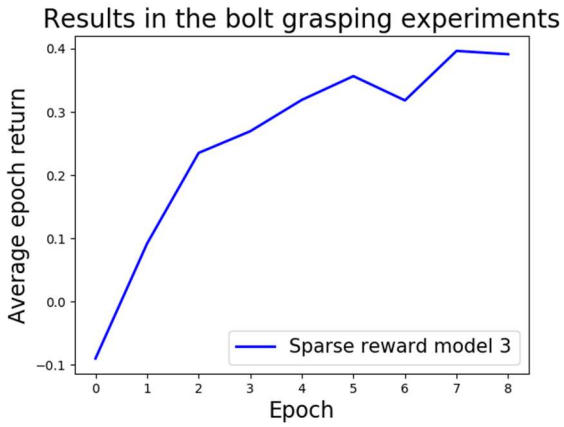
Figure. 8: Simulation training results for the experimental group in the bolt gripping experiment



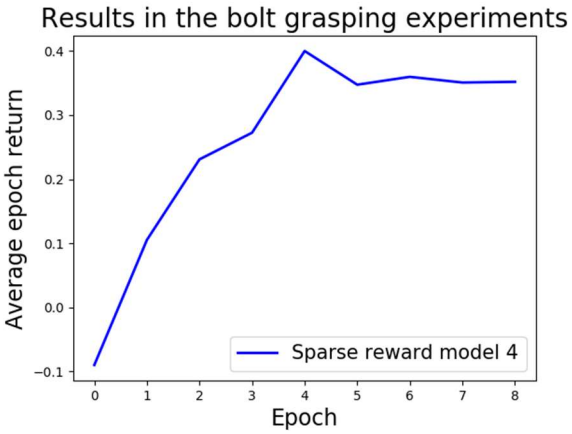
(a) Model 1 with the sparse reward.



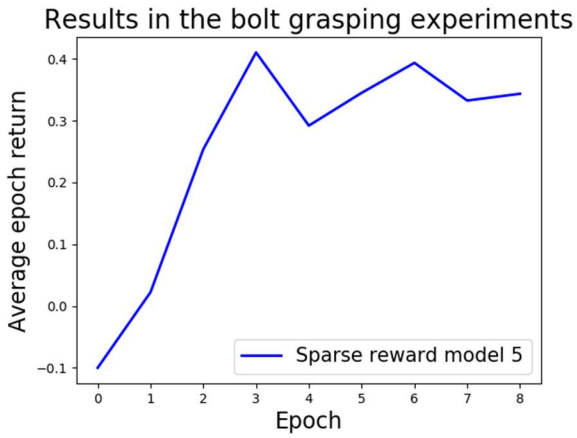
(b) Model 2 with the sparse reward



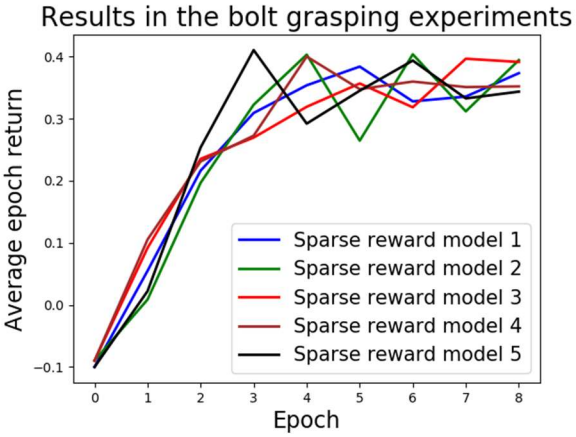
(c) Model 3 with the sparse reward.



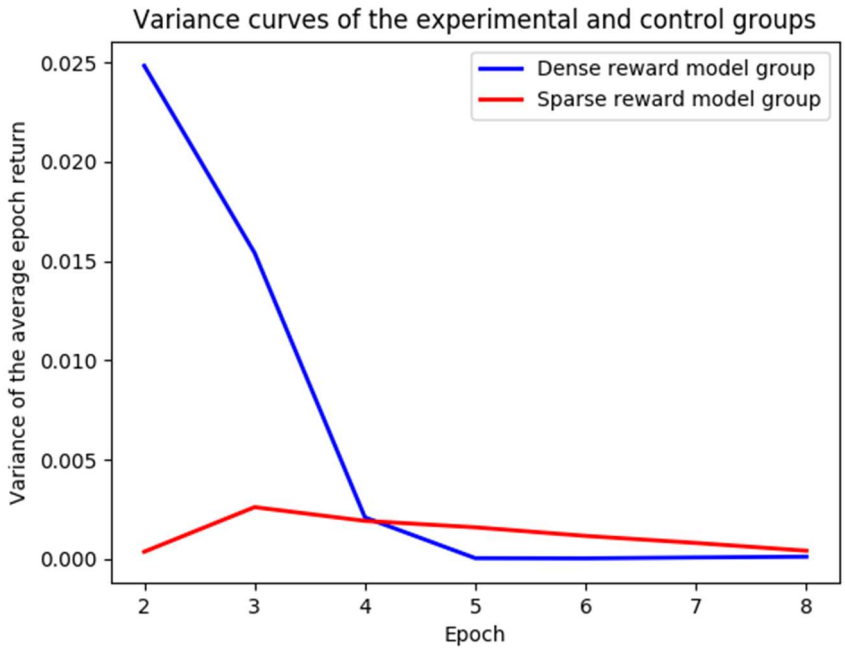
(d) Model 4 with the sparse reward



(e) Model 5 with the sparse reward



(f) Comparison of 5 models in the sparse reward



(g) Comparison of variance value graphs for the experimental and control groups.

Figure. 9: Simulation training results for the control group in the bolt gripping experiment and comparison of the variance of the results for the two main groups

Table V

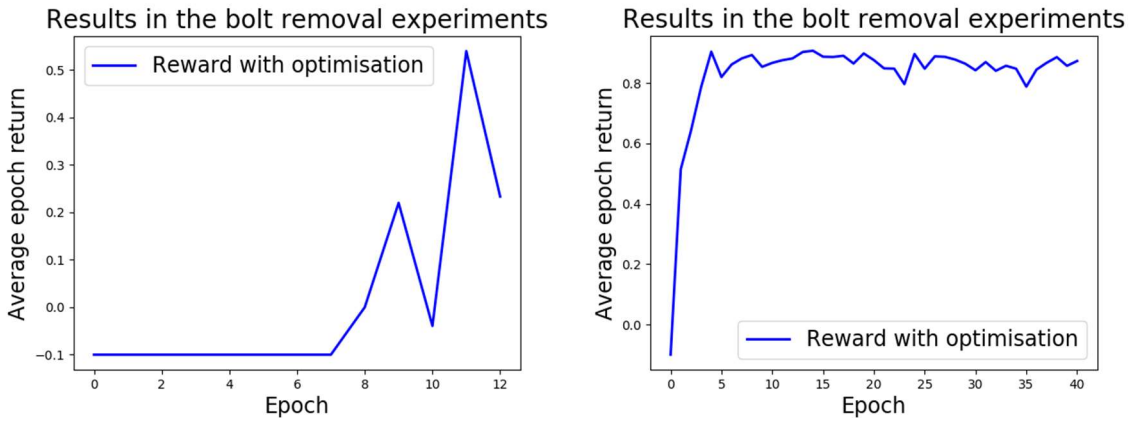
Variance values for points in each Epoch of the experimental and control groups.

No.	2	3	4	5	6	7	8
Variances for the experimental groups	0.419	2.483 $\times 10^{-2}$	1.541 $\times 10^{-2}$	2.093 $\times 10^{-3}$	4.057 $\times 10^{-5}$	3.143 $\times 10^{-5}$	7.817 $\times 10^{-5}$
Variances for the control groups	1.421 $\times 10^{-3}$	3.653 $\times 10^{-4}$	2.610 $\times 10^{-3}$	1.918 $\times 10^{-3}$	1.594 $\times 10^{-3}$	1.161 $\times 10^{-3}$	8.070 $\times 10^{-4}$

3.2. Results of the bolt removal experiment

In the bolt removal experiment, the animated simulation showed that the robot, with the initial reward function set, wandered for a long time in a very small area and never reached its goal. The improved reward function was able to drive the robot to efficiently move the bolt along a better path to successfully reach the other end of the slot and remove it. In addition, the training results of the algorithm are shown in Figure 10. Figure (a) shows the model trained according to the initial sparse reward function, where the reward values do not converge after a period of training time. In contrast, Figure b shows the successful convergence of the model data trained using the dense reward function.

.



(a) Model with the sparse reward *(b) Model with the dense reward*

Figure. 10: Data results from the simulation training in the bolt removal experiment

4. Discussion

This study verifies the feasibility of the robot autonomously learning how to disassemble the door chain model through two stages of robot simulation experiments and the introduction of deep reinforcement learning, and the help of multidimensional sensors. In addition, by comparing the simulation and training results of multiple sets of experiments with different reward functions, the sparsity of the reward function was found to be instructive for the robot's autonomous learning and essential for the stability of its autonomous learning system. The experimental results are discussed in the following section from different perspectives.

Even though the experimental group had one reward function set differently from the control group in the bolt grasping experiment, the experimental results showed that they both

completed the task better. As expected, this result shows that the robot was able to learn to complete the door chain removal task autonomously in the simulation environment, which means that the density of the reward function did not affect whether the robot could complete this result of autonomous learning. It is worth noting that because of the different reward function settings, the amount of variation in the data and the convergence values of the experimental and control groups cannot be directly compared. To further explore the effect of the sparsity of the reward function on the autonomous learning process of the robot, the variance size was calculated for the data obtained from the two groups of multiple experiments. For those variances, the variance between the data from the experimental group with the denser reward function was stable at around 0.00004 in training after Epoch 4, which was smaller than and about one-hundredth of the variance of 0.001 for the control group. This indicates that throughout multiple uses of the algorithmic model, the autonomous learning process of the robot obtained using the algorithmic model with relatively dense rewards was essentially the same each time, and therefore the results were more stable.

In contrast, in the bolt removal experiments, the control group with a sparse reward function under the same conditions showed no tendency to converge in finite time. In contrast, the experimental group with a dense reward function eventually converged at around 0.1 and performed this learning task better. Thus, the results of the experiments that successfully introduced distance rewards also suggest that the sparsity of the reward function may be a guide for the agent to learn to complete this task. However, unlike the first experiment, the reward function system in the second experiment was complicated by the introduction of rewards for multiple force conditions. Thus, the results of autonomous robot learning cannot be attributed to the sparsity of individual reward functions but rather to the system of reward mechanisms as a whole. However, whether it is linear or non-linear for the reward function that impacts the training results of the robot and how to set an upper limit on the sparsity of its application is a problem for which it is difficult to find the best solution [31]. This is likewise the reason why many scholars generally find it difficult to design good reward functions for humans and turn to the learning from demonstrations approach (LFD) and reverse reinforcement learning algorithms [23], [32].

In fact, this particular condition of the reward function has a vital role in the robot simulation process. This is because it appears most of the time when the agent is not out and when they are not successful. Unlike the constant value of 0 in the sparse reward function, the more intensive reward function sets the increase in reward value to be inversely proportional to the relative distance to the target. The former is a better guide for the robot on how to choose the right path to produce a satisfactory distance different than the latter, which is arbitrary in its path choice. Thus the denser reward function helps the robot to choose the correct path direction with more confidence.

However, there may be limitations if the reward function is too dense. For example, in Experiment 2, taking multiple complex reward functions into account under conditions where the robot's arm is not out of bounds and incomplete learning may cause the robot to mistakenly believe that the highest value can be reached in the current clamped and stationary environment, resulting in the final policy converging to a local optimum and negatively impacting learning [31]. Provided that the task requirements can be met, sparse reward mechanisms like the one in Experiment 1 are still strongly promoted because they do not require artificially designed reward functions.

Overall, the denseness of the reward function thus helps the robot train a more stable system for autonomous learning but is not directly related to the robot's ability to complete the autonomous learning task. In particular, it is crucial to avoid the limitations of artificially designed reward functions when studying the effects of their denseness in order not to mislead the experimental results.

5. Conclusion

This paper uses a deep reinforcement learning simulation to virtually implement a robot that learns autonomously and completes a door chain removal task. The experiment demonstrates the feasibility of the robot learning system designed in this paper. The sparsity of the reward function positively impacts the stability of the robot learning system by comparing the repeatability of the experimental data with that of the control group. In addition, the experiment explores the limitations of artificially set reward functions.

To further explore the practical application possibilities of this simulation model based on this experiment, the accuracy of the robot motion can be increased by order of magnitude. Also, to further investigate the effect of reinforcement learning reward functions on the development of robot autonomy, the number of experimental training samples can be increased. The effect of reward functions on properties other than the stability of the agent's autonomous learning system can be explored by setting a wider variety of reward functions ranging from sparse to dense.

6. Reference

[1] M. A. Lee *et al.*, 'Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks', *ArXiv190713098 Cs*, Jul. 2019, Accessed: Dec. 03, 2021. [Online]. Available: http://arxiv.org/abs/1907.13098

[2] Z. Zhou, P. Ni, X. Zhu, and Q. Cao, 'Compliant Robotic Assembly based on Deep Reinforcement Learning', in *2021 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, Jul. 2021, pp. 6–9. doi: 10.1109/MLISE54096.2021.00009.

[3] Z. Hou, H. Dong, K. Zhang, Q. Gao, K. Chen, and J. Xu, 'Knowledge-Driven Deep Deterministic Policy Gradient for Robotic Multiple Peg-in-Hole Assembly Tasks', in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2018, pp. 256–261. doi: 10.1109/ROBIO.2018.8665255.

[4] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, 'Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep Reinforcement Learning Approach', *Appl. Sci.*, vol. 10, no. 19, p. 6923, Oct. 2020, doi: 10.3390/app10196923.

[5] H. Poschmann, H. Brüggemann, and D. Goldmann, 'Disassembly 4.0: A Review on Using Robotics in Disassembly Tasks as a Way of Automation', *Chem. Ing. Tech.*, vol. 92, no. 4, pp. 341–359, 2020, doi: 10.1002/cite.201900107.

[6] 'Learning and generalising object extraction skill for contact-rich disassembly tasks: an introductory study | SpringerLink'. https://link.springer.com/article/10.1007/s00170-021-08086-z (accessed Dec. 08, 2021).

[7] R. Li *et al.*, 'Unfastening of Hexagonal Headed Screws by a Collaborative Robot', *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1455–1468, Jul. 2020, doi: 10.1109/TASE.2019.2958712.

[8] J. Huang *et al.*, 'An experimental human-robot collaborative disassembly cell', *Comput. Ind. Eng.*, vol. 155, p. 107189, May 2021, doi: 10.1016/j.cie.2021.107189.

[9] C. Klas *et al.*, 'The KIT Gripper: A Multi-Functional Gripper for Disassembly Tasks', in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 715–721. doi: 10.1109/ICRA48506.2021.9561336.

[10] F. J. Ramírez, J. A. Aledo, J. A. Gamez, and D. T. Pham, 'Economic modelling of robotic disassembly in end-of-life product recovery for remanufacturing', *Comput. Ind. Eng.*, vol. 142, p. 106339, Apr. 2020, doi: 10.1016/j.cie.2020.106339.

[11] B. Chen, W. Xu, J. Liu, Z. Ji, and Z. Zhou, 'Robotic Disassembly Sequence Planning Considering Robotic Collision Avoidance Trajectory in Remanufacturing', in *2020 IEEE*

18th International Conference on Industrial Informatics (INDIN), Jul. 2020, vol. 1, pp. 494–501. doi: 10.1109/INDIN45582.2020.9442129.

[12] T. Johannink *et al.*, 'Residual Reinforcement Learning for Robot Control', *ArXiv181203201 Cs*, Dec. 2018, Accessed: Dec. 03, 2021. [Online]. Available: http://arxiv.org/abs/1812.03201

[13] S. James and E. Johns, '3D Simulation for Robot Arm Control with Deep Q-Learning', *ArXiv160903759 Cs*, Dec. 2016, Accessed: Dec. 05, 2021. [Online]. Available: http://arxiv.org/abs/1609.03759

[14] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, 'Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control', in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 7512–7519. doi: 10.1109/IROS51168.2021.9635857.

[15] 'Production - Franka Emika'. https://www.franka.de/production (accessed May 14, 2022).

[16] 'Convex hull - MATLAB convhull'. https://www.mathworks.com/help/matlab/ref/convhull.html (accessed May 14, 2022).

[17] R. Michalík and A. Janota, 'The PyBullet module-based approach to control the collaborative YuMi robot', in *2020 ELEKTRO*, May 2020, pp. 1–4. doi: 10.1109/ELEKTRO49696.2020.9130233.

[18] G. Singh and M. Sachan, 'Multi-layer perceptron (MLP) neural network technique for offline handwritten Gurmukhi character recognition', in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, Dec. 2014, pp. 1–5. doi: 10.1109/ICCIC.2014.7238334.

[19] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, 'Reinforcement Learning with Augmented Data'. arXiv, Nov. 05, 2020. Accessed: May 14, 2022. [Online]. Available: http://arxiv.org/abs/2004.14990

[20] D. Yarats, I. Kostrikov, and R. Fergus, 'Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels', presented at the International Conference on Learning Representations, Sep. 2020. Accessed: May 14, 2022. [Online]. Available: https://openreview.net/forum?id=GY6-6sTvGaf

[21] Y. He and S. Liu, 'Analytical Inverse Kinematics for Franka Emika Panda – a Geometrical Solver for 7-DOF Manipulators with Unconventional Design', in *2021 9th International Conference on Control, Mechatronics and Automation (ICCMA)*, Nov. 2021, pp. 194–199. doi: 10.1109/ICCMA54375.2021.9646185.

[22] P.-W. Chou, D. Maturana, and S. Scherer, 'Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution', in *Proceedings of the 34th International Conference on Machine Learning*, Jul. 2017, pp. 834–

843. Accessed: May 14, 2022. [Online]. Available: https://proceedings.mlr.press/v70/chou17a.html

[23] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, 'Overcoming Exploration in Reinforcement Learning with Demonstrations', in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6292–6299. doi: 10.1109/ICRA.2018.8463162.

[24] 'Part 2: Kinds of RL Algorithms — Spinning Up documentation'. https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#links-to-algorithms-in-taxonomy (accessed May 14, 2022).

[25] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, 'Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning'. arXiv, Dec. 01, 2017. Accessed: May 14, 2022. [Online]. Available: http://arxiv.org/abs/1708.02596

[26] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, 'Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning'. arXiv, Feb. 28, 2018. Accessed: May 14, 2022. [Online]. Available: http://arxiv.org/abs/1803.00101

[27] V. Mnih *et al.*, 'Asynchronous Methods for Deep Reinforcement Learning'. arXiv, Jun. 16, 2016. Accessed: May 14, 2022. [Online]. Available: http://arxiv.org/abs/1602.01783

[28] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, 'Policy Gradient Methods for Reinforcement Learning with Function Approximation', in *Advances in Neural Information Processing Systems*, 1999, vol. 12. Accessed: May 14, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, 'Proximal Policy Optimization Algorithms', *ArXiv170706347 Cs*, Aug. 2017, Accessed: Dec. 05, 2021. [Online]. Available: http://arxiv.org/abs/1707.06347

[30] L. Martino, V. Elvira, J. Míguez, A. Artés-Rodríguez, and P. M. Djurić, 'A Comparison Of Clipping Strategies For Importance Sampling', in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, Jun. 2018, pp. 558–562. doi: 10.1109/SSP.2018.8450722.

[31] M. Vecerik *et al.*, 'Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards'. arXiv, Oct. 08, 2018. Accessed: May 14, 2022. [Online]. Available: http://arxiv.org/abs/1707.08817

[32] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada, 'Hybrid Trajectory and Force Learning of Complex Assembly Tasks: A Combined Learning Framework', *IEEE Access*, vol. 9, pp. 60175–60186, 2021, doi: 10.1109/ACCESS.2021.3073711.