Learning of Long-Horizon Sparse-Reward Robotic Manipulator Tasks With Base Controllers

Guangming Wang, Graduate Student Member, IEEE, Minjian Xin, Wenhua Wu, Zhe Liu, and Hesheng Wang, Senior Member, IEEE

Abstract—Deep reinforcement learning (DRL) enables robots to perform some intelligent tasks end-to-end. However, there are still many challenges for long-horizon sparse-reward robotic manipulator tasks. On the one hand, a sparse-reward setting causes exploration inefficient. On the other hand, exploration using physical robots is of high cost and unsafe. In this article, we propose a method of learning long-horizon sparse-reward tasks utilizing one or more existing traditional controllers named base controllers in this article. Built upon deep deterministic policy gradients (DDPGs), our algorithm incorporates the existing base controllers into stages of exploration, value learning, and policy update. Furthermore, we present a straightforward way of synthesizing different base controllers to integrate their strengths. Through experiments ranging from stacking blocks to cups, it is demonstrated that the learned state-based or image-based policies steadily outperform base controllers. Compared to previous works of learning from demonstrations, our method improves sample efficiency by orders of magnitude and improves performance. Overall, our method bears the potential of leveraging existing industrial robot manipulation systems to build more flexible and intelligent controllers.

Index Terms—Base controllers, deep reinforcement learning (DRL), long-horizon sparse reward, robotic manipulator tasks.

I. INTRODUCTION

The research of learning-based control is developing rapidly [1], [2], [3], [4]. Among them, deep reinforcement learning (DRL) has been applied extensively to complicated decision-making problems in the past few years, including board games like Go [5] and video games like StarCraft [6]. Meanwhile, a lot of research has been done on applying DRL to robot control problems [7], [8], [9], [10], [11], [12], [13]. However, DRL algorithms are often data-hungry, requiring millions of interactions with the environment to learn a single task. Exploration is even more inefficient when the reward is sparse and delayed, which is a common situation in robot tasks. Besides, the random exploration of a robot in the real world is of high cost and unsafe. Although there is a scheme of training in simulation and testing in a real environment, the gap between simulation and real environment always exists, which requires a high degree of verisimilitude in the simulation environment.

For the long-horizon sparse-reward robotic manipulator tasks, the robot needs to perform correct actions continuously and finally complete the task to obtain the reward. At the beginning of training, it is almost impossible to randomly explore a series of corrective actions to complete the task. As a result, it is hard for the robot to obtain rewards, so the strategy cannot be optimized, or the optimization is very slow.

Previous works have used demonstrations [14], [15], [16], [17], [18] to speed up DRL. Although this method could successfully learn some long-horizon sparse-reward tasks, it is still not a perfect solution to the efficiency problem. Learning tasks like “pick-and-stack” still take millions of interactions with the environment. Furthermore, additional hardware like virtual reality equipment [14], [19] or 3-D motion controllers [15], [17] are required to collect demonstrations. Virtual reality equipment ensures that humans and robots share the same observation and action space. The 3-D motion controllers allow the manipulator to be operated by position control. The demonstration-based approach implements guidance by calculating the behavior cloning (BC) loss on the demonstration dataset. The demonstration dataset is generally small, and BC loss does not play a role outside the demonstration dataset, so the guidance effect is limited. In addition, at the beginning of training, a large number of ineffective actions explored by the agent will be put into the replay buffer. This makes the agent learns slowly.

Residual reinforcement learning (RL) [20], [21] introduces a baseline in the learning process of the network controller. However, it only learns the correction of the base controller. Its effect is limited by the base controller, so it is difficult to explore a better strategy. It is also difficult to find appropriate compensation for sparse rewards, which makes the overall effect of the base controller and network controller still poor.
Moreover, it cannot combine the advantages of multiple base controllers.

Considering these, we propose a hybrid method for robotic task learning with existing controllers, which is referred to as base controllers. The base controller has its own fixed mode to guide the network to learn. In addition, conservative and slow controllers with high accuracy and aggressive and efficient controllers with low accuracy are difficult to be selected as appropriate guiding baselines. Therefore, we also study an ensemble strategy of base controllers, to merge the advantages of different base controllers. The network can learn a flexible strategy with high efficiency and high precision. The main contributions include as follows.

1) Long horizons and sparse rewards make it difficult for robots to obtain rewards to optimize networks. We present a novel learning scheme that incorporates base controllers to resolve the challenges of sparse-reward and long-horizon robotic tasks. The proposed approach greatly relieves the exploration cost during learning.

2) We extend our approach to an ensemble of base controllers so that the network can aggregate the advantages of these controllers. The final network performance is better than all these base controllers.

3) The experiments demonstrate that, compared with learning from demonstrations, our approach accelerates the learning speed by orders of magnitude and is also effective in handling difficult tasks which cannot be solved by demonstration learners.

II. RELATED WORK

A. Imitation Learning

Imitation learning (IL) is broadly adopted in robotics, ranging from automated driving [22], [23] to unmanned aerial vehicle [24] control. The simplest way of doing IL is BC [22], which intends to match the actions of the learner to those of an expert. BC is not safe due to overfitting and distribution drift issues. IL algorithms like dataset aggregation (DAGGER) [25], [26] address this by collecting data with an annealed ratio of expert control, which helps achieve low-cost learning or regret minimization. Nevertheless, a huge amount of expert data needs to be collected. Recent algorithms can also learn implicitly from demonstration data, such as inverse RL (IRL) [27] and generative adversarial IL (GAIL) [28].

B. Reinforcement Learning

State-of-the-art DRL algorithms have been applied in the robot learning field, ranging from robotic manipulator control using deep Q-network (DQN) [29], [30], dexterous robot hand manipulation using proximal policy optimization (PPO) [10], [31], up to applications of deep deterministic policy gradient (DDPG) [32], [33], and soft actor-critic (SAC) [34]. For simple sparse-reward settings, hindsight experience replay (HER) [35] proposes the notion of goal-conditioned tasks. But HER is hard to be applied to more complex tasks with bottlenecks [17]. DivAC [36] extends the divergent strategy iterative algorithm to the continuous action space problem, which can deal with the tradeoff between exploration and exploitation. Yang et al. [37] propose the universal option framework (UOF) to solve the multistep tasks by training symbol planning and motion control strategies simultaneously.

Some works investigate model uncertainty in RL algorithms [38], [39], [40]. Kahn et al. [38] propose a model-based learning algorithm combining uncertainty perception and control method. Lütjens et al. [39] use dropout and bootstrapping to train a set of LSTM networks for model uncertainty estimation and construct a safe RL framework. Wu et al. [40] propose an uncertainty weighted actor-critic (UWAC) to deal with the uncertainty in offline RL.

C. Combining Imitation With Reinforcement Learning

Researchers have been trying to combine IL with RL to get the best of both worlds. DDPGGfD [41] puts demonstration in the replay buffer of DDPG to learn peg insertion tasks. Nair et al. [14] augment DDPG with a Q-filtered BC loss to learn with a separate demonstration replay buffer. In our approach, all transformations are stored in the replay buffer of DDPG, and there is no distinction between whether the actions in the transformation are decided by the base controller or the agent. The model is more concise. We gradually reduce the possibility of taking a base controller by a decaying coefficient. Our BC loss is similar to [14]. The BC loss of [14] is only calculated in the demonstration. However, we calculate the BC loss when the action of the actor network is inferior to the base controller. This allows the agent to learn from the base controller more quickly and gradually reduce the dependence on the base controller.

Zhu et al. [15] employ GAIL to learn from demonstrations and use state prediction as auxiliary tasks to learn a deep visuomotor policy for a robot hand. Ding et al. [17] use demonstrations to overcome bottlenecks in HER. Hermann et al. [16] sample demonstrations according to a curriculum of increasing difficulty to help learn a visuomotor policy. All of the above works use demonstrations to realize robot learning, but tasks like “pick-and-stack” still take millions of environmental interactions. Furthermore, additional hardware is required to collect demonstrations. Finally, the agent is prone to making mistakes in the initial stage, which is unfavorable when mistake costs are expensive. In contrast to learning from demonstrations, learning with existing controllers often directly uses a controller to aid exploration. Guided policy search (GPS) [42], [43] optimizes a model-based controller and then learns a policy in a supervised manner. Residual RL [20], [21] learns a compensation policy that is strongly dependent on the existing controller, while our method learns a complete policy and the base controller can be discarded once learning is done. Our method can also utilize an ensemble of base controllers, while residual RL can only use one controller.

III. METHOD

We formulate the Markov decision process (MDP) as standard RL. For state set $S$ and action set $A$ in environment $E$, a policy $\mu$ is a mapping $S \rightarrow A$, and $P : S \times A \rightarrow S$ is the transition probability. For every time step $t$, the agent receives
observation \(s_t\), takes an action \(a_t\), and then receives next observation \(s_{t+1}\) and reward \(r_t\). The goal is to maximize the accumulated discounted reward \(R_t = \sum_{t=0}^{T} \gamma^{t-t} r_t\), where \(\gamma\) is the discount factor. The optimal \(Q\) value is defined as the expected reward after taking an action from a state and acting optimally thereafter: \(Q^*(s, a) = \max_{\pi} \mathbb{E}_{s_t, a_t \sim \pi} [R_t | s_t, a_t] \). A base controller is denoted as \(\mu_b(s)\) and can be queried for actions regarding arbitrary states. Our algorithm builds on DDPG [32], a standard actor-critic algorithm for continuous control which is often preferred in previous works [14], [41] for its utilization of off-policy data. The actor network is denoted as \(\mu(s|\theta^\mu)\), and the critic network is denoted as \(Q(s,a|\theta^Q)\), where \(\theta^\mu\) and \(\theta^Q\) are corresponding network weights. In the sections below, we will describe our algorithm, DDPG with base controllers (DDPGwB), as shown in Fig. 1, including methods during exploration, value estimation, and policy update to incorporate a base controller. We will also show how the algorithm scales from one base controller to an ensemble of base controllers.

### A. Mixed Q-Control

Exploration is challenging in robotic tasks, especially when the reward is sparse. By utilizing a base controller to explore in the initial stage and linearly decaying its control ratio, the agent is able to experience a regular positive reward. The agent also avoids unsafe states and unnecessary explorations. However, simply annealing the ratio could result in abandoning the base controller too early or too late. When abandoning too early, the agent might crumble due to insufficient guidance, and when abandoning too late, the agent might not outperform the base controller since it cannot explore on its own. To address this, a novel mechanism is proposed that allows the agent to flexibly adjust the control ratio. Specifically, the probability that an agent must execute the base controller’s action, \(\epsilon\), is initialized to 1 and decreases by \(\delta\) for every environment step. With \(1 - \epsilon\) probability, the agent selects the best action based on the prediction of the critic network

\[
a_t = \arg\max_{a \in B_t \cup \{\mu(s)\}} Q(s_t, a|\theta^Q) .
\]

The intuition behind this is that in Q-learning [44], the learned policy is selecting the action with the greatest value over all possible actions given an arbitrary state. Since the critic network in DDPG performs the same role as the Q-network in DQN, it is natural to adopt the same fashion of choosing actions via \(\arg\max\).

### B. Base Controller Bootstrap

In the update step of the critic network, DDPG, like many other RL algorithms, uses the Bellman equation

\[
Q^*(s, a) = \mathbb{E}_{s_t, a_t \sim \pi} [r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})]
\]

(2)

to estimate the \(Q\) value, where “\(\pi\)” denotes the optimal policy. Unlike Q-learning, DDPG deals with continuous actions and thus loses the ability to efficiently compute the maximum \(Q\) value of the next state. Hence, DDPG uses the \(Q\) value of its current policy. In practice, target networks (indicated with “\(\bar{\cdot}\)”) below are used for stabilized learning, whose weights are slowly updated. We use \(y_t\) to represent the bootstrap target

\[
y_t = r_t + \gamma Q^*(s_{t+1}, \mu'(s_{t+1}|\theta^\mu)|\theta^Q) .
\]

(3)

where \(\mu'(s|\theta^\mu)\) denotes the target actor network, \(Q^*(s, a|\theta^Q)\) denotes the target critic network, \(\theta^\mu\) is the weights of the
Update the target networks:

1. Execute action
2. Update critic by minimizing the Bellman error;
3. Store transition

Due to inaccurate value estimates and a deterministic policy, such unbounded update tends to be very brittle as discussed in [45], as the actor network will often try to exploit peaks in the Q-function and diverge. To relieve this, we apply BC with respect to the base controller as a regularization tool that guides the gradient update direction

\[ \nabla_{\theta^\pi} L_a = -\nabla_{\theta^\pi} L_{BC} + \lambda \nabla_{\theta^\pi} J \]

where \( \lambda \) is a coefficient balancing two losses. Specifically, the BC gradient is imposed only on actions that are worse than those of the base controller as decided by the critic network. Note that this does not violate the optimization goal of maximizing the accumulated reward. This kind of filtering is similar to [14], except in [14] BC loss is imposed on a demonstration buffer.

D. Ensemble of Base Controllers

For complex tasks, there is sometimes no good base controller, but some weak controllers can be easily designed. These controllers have their own characteristics: some high efficiency but low precision, some high precision but low efficiency. Then, it is ideal to learn a policy from an ensemble of multiple base controllers and integrate their strengths. To this end, our algorithm easily scales to multiple base controllers. Considering \( K \) base controllers: \( \mu_b(s), \mu_b(s), \ldots, \mu_b(s) \), we define the ensemble base controller action set as

\[ \mathcal{B}_t = \{ \mu_b(s_t), \mu_b(s_t), \ldots, \mu_b(s_t) \}. \]

With probability \( \epsilon \), the action is randomly selected from \( \mathcal{B}_t \). With probability \( 1 - \epsilon \), the choice is made amount actions in \( \mathcal{B}_t \) and the actor network action according to the modified (1)

\[ a_t = \arg \max_{a \in \mathcal{B}_t \cup \{ \mu(s_t) \}} Q(s_t, a|\theta^\pi). \]

The bootstrap target form in value learning now also extends from (4) to

\[ y_t = r_t + \gamma \max_{a \in \mathcal{B}_t \cup \{ \mu(s_t) \}} \left[ Q(s_{t+1}, a|\theta^\pi) - \mu(s_t|\theta^\pi) \right]. \]

Finally, \( L_{BC} \) in (6) becomes

\[ L_{BC} = \frac{1}{N} \sum_{i=1}^{N} \left[ \mu(s_t|\theta^\pi) - \mu(s_t|\theta^\pi) \right]^2. \]

Using an ensemble of base controllers helps the agent to avoid mistakes caused by a single controller during exploration, as in (8). It also helps the critic network to further benefit from diverse strategies, as reflected in the bootstrap target form in (9). In turn, those learned values help the agent shape a better policy as in (10). In this way, even if the individual base controller is weak, a learned policy absorbs the strengths in each of them and has a good performance.

The complete pseudo-code is presented in Algorithm 1. We unified exploration, value learning, and policy learning into a single framework to take advantage of one or more base controllers. Base controller bootstrap helps the critic to

---

**Algorithm 1 DDPG With Base Controllers (DDPGwB)**

**Input:** actor network \( \mu(s|\theta^\pi) \), critic network \( Q(s, a|\theta^\Omega) \), target network \( \mu^\prime \) and \( Q^\prime \), \( K \) base controllers: \( \mu_b(s), \mu_b(s), \ldots, \mu_b(s) \), replay buffer \( R \), Gaussian noise \( \mathcal{N} \)

**Output:** learned policy \( \mu(s|\theta^\pi) \)

1. \( \epsilon = 1; \)
2. for \( \text{episode} = 1, M \) do
3. \( \text{Get initial observation } s_1; \)
4. for \( t = 1, T \) do
5. if random < \( \epsilon \) then
6. \( a_t = \mu_b(s_t) \) random \( B_t; \)
7. else
8. \( \text{Get } a_t \text{ according to Equation (8); } \)
9. \( a_t \leftarrow a_t + N; \)
10. \( \text{Execute action } a_t, \text{ observe reward } r, \text{ and next state } s_{t+1}; \)
11. \( \text{Store transition } (s_t, a_t, r_t, s_{t+1}) \text{ in } R; \)
12. \( \epsilon \leftarrow \epsilon - \delta; \)
13. \( \text{Sample a random minibatch of } N \text{ transitions } (s_t, a_t, r_t, s_{t+1}) \text{ from } R; \)
14. \( \text{Get } y_t \text{ according to Equation (9); } \)
15. \( \text{Update critic by minimizing the Bellman error; } \)
16. \( \text{Update actor according to Equation (6); } \)
17. \( \text{Update the target networks: } \)
18. \( \theta^\mu \leftarrow \tau \theta^\mu + (1 - \tau) \theta^\mu, \theta^Q \leftarrow \tau \theta^Q + (1 - \tau) \theta^Q; \)

---

The policy update in DDPG is done by back-propagating through the critic network

\[ \nabla_{\theta^\pi} J = \frac{1}{N} \sum_{i} \nabla_{\theta^\pi} Q(s_i, a_i|\theta^\pi) \nabla_{\theta^\pi} \mu(s_i|\theta^\pi). \]
avoid suffering from a terrible actor, so the critic network in charge of both exploration and policy update can be learned better. The brittleness of the actor network is greatly eased, as it will be shown in experiments that abandoning the use of the target actor network does not create drastic decay in our algorithm’s performance, which is unusual for DDPG.

IV. EXPERIMENTS

In this section, we provide a description of the tasks we evaluate our algorithm on. Then the network architectures are described, along with a straightforward template for designing base controllers. Finally, we give a quantitative evaluation of our algorithm in terms of learned policies’ success rates, learning efficiency, and the ability to utilize an ensemble of base controllers. And we compare it to previous methods.

A. Task Description

We use PyBullet [46] simulator and a KUKA robotic manipulator for our experiments. The output action of the base controller and the actor network is \( a_t = [x, y, z, e, f] \), where \( x, y, z \) are the change in the end-effector position, \( e \) is the change in the end-effector yaw angle, and \( f \) is the change in the gripper open angle. They have a range of \([-1, 1]\). The observation consists of proprioceptive data of the robotic manipulator and either state information of two objects or images. The proprioceptive data of the robotic manipulator is 8-D, including end-effector position, orientation, gripper open angle, and the force on gripper fingers. The state information of two objects is 12-D, including positions and orientations. The images are binocular RGB images from two sides of the table, with a resolution of 128 \( \times \) 128. Our algorithm is verified under the following three tasks.

1) Stacking: This task involves two cubes initialized at random positions and orientations on the table, each with a side length of 5 cm. The goal is to stack the green cube onto the purple cube.

2) Block–Cup: This task involves a cube and a cup initialized at random positions and orientations on the table. The cube has a side length of 5 cm. The cup’s height is 15 cm and has a square opening with a side length of 6 cm. The goal is to place the green cube into the blue cup while not tipping the cup over.

3) Cup–Cup: This task involves two cups initialized at random positions and orientations on the table. The green cup’s opening has a side length of 5 cm. The blue cup’s opening has a side length of 6 cm. Both cups’ height is 15 cm. The goal is to place the green cup into the blue cup while not tipping either cup over.

Screenshots of all tasks are presented in Fig. 2. In all tasks, we employ a truly sparse reward, that is, a reward of 1 if the goal is achieved, and a reward of 0 for other scenarios, as follows:

\[
  r_t = \begin{cases} 
    1, & \text{the goal is achieved} \\
    0, & \text{otherwise.} 
  \end{cases}
\]  

An episode is terminated if it reaches 100 steps or the agent achieves the goal. In block–cup or cup–cup tasks, an episode is also terminated when a cup is tipped, and the reward is 0. All tasks are designed to reflect the challenges of sparse-reward and long-horizon robotic tasks, that is, the robot must accomplish a sequence of actions before receiving a proper reward signal. For example, for the cup–cup task, the robot must first reach for the green cup, grab the cup, then align the green cup with the blue cup, and finally place it into the blue cup to obtain the reward.

We highlight the increasing difficulties in these tasks: First, the agent has to pick up the object. Second, it cannot tip the cup over when moving the object. Finally, it has to match the object precisely with the opening. The risk of tipping over the cup makes the tasks significantly harder and correspond to a kind of safety requirement, for example, fragile things are not allowed to be knocked over. A proper learning scheme should avoid making those expensive mistakes.

B. Network Structure

1) State-Input: The actor network receives 20-D input and outputs 5-D action, with two hidden layers each having 256 units and ReLU activation. Activation of the last layer is Tanh. The critic network receives 25-D input and outputs 1-D \( Q \) value, with two hidden layers each having 256 units and ReLU activation. Activation of the last layer is Sigmoid.

2) Image-Input: The normalized image first passes through a convolutional neural network (CNN) with three convolution layers of \( 3 \times 3 \) kernel size, one stride, 8-channel output, and ReLU activation, then passes through channel-wise spatial softmax [42] to get feature points (16-D). The binocular images share one CNN, and their output features are concatenated, resulting in a 32-D feature output. This image feature concatenated with proprioceptive data is then passed through a fully connected network same as the state-input actor

Fig. 2. Screenshots of all tasks in different stages described in Section IV-A. A screenshot of the cup–cup task depicts a failed trial, where the blue cup is tipped over by the robot. (a) Stacking. (b) Block–cup. (c) Cup–cup.
network, except for different input dimensions. A state-input critic is coupled with an image-input actor during training as in [15]. Note that the use of state information here is for learning only, and the trained image-based policies work without state information. Fig. 3 outlines the entire neural network architecture.

C. Base Controllers

Given state information and proprioceptive data, we provide a simple controller template in Algorithm 2 that covers a variety of tasks. \( p_{\text{desired}} \) represents the desired end-effector (object) position and yaw angle. \( p_{\text{curr}} \) represents the current end-effector (object) position and yaw angle. The controller first tries to reach and grasp an object. Once the gripper holds the object, the controller will place it in the goal position. We use simple proportional controllers to calculate actions for the end-effector. The proportionality coefficient \( K_p \) is 5 for position and 2 for angular actions. Force on gripper fingers is used to tell whether there is a successful grasp. The experiments show that Algorithm 2 produces decent success rates.

D. Training Details

We use the Adam optimizer and a learning rate of \( 10^{-3} \) for both actor and critic networks. Exploration noise during training is Gaussian distribution with 0.1 standard deviation and 0 mean. The discount factor \( \gamma \) is 0.99. The soft update parameter \( \tau \) for the target networks is \( 5 \times 10^{-3} \). The batch size is 256, and the replay buffer size is \( 10^5 \). We set the loss balance coefficient of the actor network \( \lambda \) in (6) as \( 2 \times 10^{-2} \). The ratio of the direct use of the base controller \( \epsilon \) is initialized to 1, and the decreased value at each step \( \delta \) is \( 2 \times 10^{-5} \). All experiments are performed on an RTX2080 Ti.

E. Results

During the training, the actor model is tested for 100 episodes every 30 training episodes. We calculate the success rate and save the model with the highest success rate. After the completion of the training, the saved model is tested for 1000 episodes (no Gaussian noise). The experiments are repeated five times, and the average of the test results is the final test performance, as listed in Table I. We also measure the training performance and the average success rate of the agent in the interaction process with the environment throughout the training process. This reflects the safety requirement of the training process. The results are the training performance in Table I. Due to the Gaussian noise in the training process and the poor performance of the agent in the early training, the performance of the training is worse than that of the test.

1) Comparison With Other Methods: We compare our method with HER [35], Residual RL [20], and the method of learning from demonstrations (DDPG w/BC + Demo) [14]. Learning curves (success rates for evaluation) with state-input are summarized in Fig. 4(a)–(c).

HER [35] uses a strategy to sample a set of additional goals for the replay buffer to improve sampling efficiency.
In our tasks, whether the robotic manipulator can successfully grasp the block is one of the bottlenecks of the task. If HER can successfully grasp the block through exploration, it can strengthen the grasping behavior and start the exploration in the next stage. However, random exploration is almost impossible due to the high complexity of the action space. In their paper, they set half training episodes to start with the block being grasped. This initialization is equivalent to artificially crossing a bottleneck and making the problem easier. However, this approach is not useful when it is difficult to find one such keyframe. Instead, our approach uses the base controller to guide agents to explore, and agents can quickly learn how to grasp. Since we use mixed Q-control to choose actions, most of the transitions in the replay buffer work, which makes our sampling efficiency much higher.

In all tasks, our method requires only about 0.2 million environment steps or less to converge, while learning from demonstrations struggles to converge in limited environment steps. In previous works, Nair et al. [14] learn a much easier pick-and-place task: the robotic manipulator has to pick up a block on the table and move it to a position in space, where neither orientation nor object interaction is considered. In their paper, about 1 million environmental steps are used to converge. According to our results, as the difficulty level of the task increases (more bottlenecks), it becomes harder for demonstration learners to find a sensible policy, and performances even start to degrade as the training progresses. For the cup–cup task, the final performance of learning from demonstrations is almost down to zero.

Residual RL [20] also uses a base controller. They learn the compensation amount of a base controller and superimpose the compensation amount on the base control action as the actual control action. Although it can improve the performance of the base controller, the policy is strongly dependent on the base controller. Therefore, other more effective policies cannot be explored, which limits performance improvement. In addition, it is difficult to explore effective compensation for random exploration, so the strategy convergence is slow. In our approach, thanks to mixed Q-control and base controller bootstrap, poor base controller performance is slowly discarded as network performance is progressively improved under guidance. Our method can learn the advantages of the base controller and explore a better strategy to replace the poor part of the base controller. This is more flexible and effective than residual RL.

To characterize the safety of learning, we present the success rates during training in Fig. 4(d)–(f), which are the performances the agents hold while actually interacting with the environment. Note that all policies’ performances are naturally lower than those during evaluation because of Gaussian noise. Results show that our method learns at a stable success rate throughout the training process. The base controller helps prevent the agent from making mistakes like tipping over the cup in the initial stage. However, learning from demonstrations starts learning with a success rate of 0. At the beginning of training, due to the random compensation of the network, the performance of residual RL is lower than that of the base controller. By contrast, our method is safer during training and more suitable for real-world training.

2) Learn From Images: We run a set of experiments that use binocular RGB images as the actor network’s input instead of state information. We train the agent for 0.4 million environment steps (50 gradient steps per episode) and evaluate the final learned policies’ performances by running 1000 test episodes for each task. The image-based policies achieve success rates of 87.28%, 68.74%, and 49.18%, respectively, for stacking, block–cup, and cup–cup tasks, proving that our method also applies to image-based tasks. The performance is worse than the state-based result, which we believe is because the feature extraction capability of the convolutional neural network is limited.

3) Abandon the Target Actor: We further validate our algorithm by abandoning the use of a target actor network, which is an unusual ablation since DDPG needs both the target actor and target critic networks for stabilized learning. The target actor network generates stable actions, and the target critic network generates stable $Q$ values according to the actions of the target network, which provides stable target points for network training and makes training stable. In our algorithm, the actions of the baseline controller and the target actor network are compared by the target critic network. The target critic network will select the optimal estimation $Q$ value. Our base controller itself is a very stable actor, and its actions further ensure the stability of the $Q$ value estimated by the target critic network. When the target action network is removed, the base control still comes into play and the training is still effective and stable. As shown in Fig. 5, abandoning the use of a target actor network does not create drastic decay to our algorithm’s performance, and policies are still stably learned outperforming base controllers, which highlights that our algorithm greatly increases the stability of actor.

| Method                      | Test performance | Training performance |
|-----------------------------|------------------|----------------------|
|                             | Stacking | Block-cup | Cup-cup | Stacking | Block-cup | Cup-cup |
| HER [35]                    | 0%       | 0%        | 0%      | 0%       | 0%        | 0%      |
| DDPG w/BC + Demo [14]       | 63.40%   | 56.90%    | 14.16%  | 15.79%   | 7.69%     | 1.67%   |
| Residual RL [20]            | 93.82%   | 70.62%    | 40.94%  | 82.82%   | 57.07%    | 21.03%  |
| Base Controller             | 89.04%   | 57.66%    | 28.14%  | 89.04%   | 57.66%    | 28.14%  |
| DDPGwb (Ours)               | 95.58%   | 84.94%    | 65.40%  | 87.13%   | 74.34%    | 45.02%  |
Fig. 4. (a)–(c) Learning curves (success rates during evaluation) of three tasks. (d)–(f) Success rates in actual training of three tasks. Solid lines represent the mean, and shaded areas represent the standard deviation. The number of environment steps is used as an x-axis label to convey sample complexity. Curves are averaged from five runs with different random seeds and smoothed for clarity. The performance of the base controller is denoted by the horizontal dashed line.

Fig. 5. Effect of abandoning the use of target actor networks. Without the target actor network, the performance of our algorithm is not significantly reduced. (a) Stacking. (b) Block–cup. (c) Cup–cup.

| Method          | Average steps | Success rate | Misbehavior rate |
|-----------------|---------------|--------------|------------------|
| Base controller |               |              |                  |
| Base 0          | 93.4          | 63.8%        | 25.9%            |
| Base 1          | 120.2         | 83.1%        | 13.2%            |
| DDPGb           |               |              |                  |
| Base 0          | 70.7          | 75.6%        | 18.0%            |
| Base 1          | 96.7          | 82.0%        | 14.0%            |
| Base 0&1        | 88.3          | 86.5%        | 9.6%             |

4) Learn With an Ensemble of Base Controllers: For an ensemble of base controllers, we design two base controllers: a base controller 0 with high efficiency but low accuracy, and a base controller 1 with high accuracy but low efficiency. When the ensemble of base controllers works, we randomly choose one of the actions to perform. First, we train our algorithm with these two base controllers. Then, we train our algorithm with the ensemble of the two base controllers. The total training step length is 1 million steps. The maximum step length of each episode is up to 140 steps to fully perform the slower base controller.

The results are shown in Fig. 6 and Table II. When the two base controllers are trained separately, the agent performance we get exceeds that of the base controller. With the combination of controllers, our agent’s performance improves even more. The success rate is higher and the error rate is lower than the two base controllers and the model trained by them, respectively, with the average completion step size between base controller 0 and base controller 1. It can be seen that when using an ensemble of base controllers, our algorithm can learn the advantages of each base controller at the same time. The performance is better than using a single controller, which reflects the superiority of our algorithm. However, residual RL [20] can only use one base controller.
5) Sim2Real Policy Transfer: In order to verify the effectiveness of the strategy in real scenarios, we test the actor network trained in the simulation on a real KUKA LBR IIWA manipulator. The real-world setup is the same as the simulation environment (including robot dynamics and object size). We use state input to test the trained actor network on the three tasks. The location and orientation of the blocks and cups are random. The initial position of the manipulator is the same each time. In ten experiments, the success rate was 90% for stacking, 70% for block–cup, and 50% for cup–cup. This shows the robustness of our algorithm and the potential for practical applications. Video demonstrations of the real-world environment are uploaded to supplemental materials.

V. CONCLUSION

In this article, we propose DDPGwB, an algorithm that utilizes base controllers to efficiently and safely learn challenging sparse-reward robot tasks. DDPGwB incorporates base controllers into stages of exploration, \( Q \) value estimation as well as policy update, and learns state-based or vision-based policies that exceed the performances of base controllers. At the same time, our algorithm can be extended to an ensemble of base controllers to concentrate the advantages of each controller. The method proposed in our research may facilitate DRL applications in industrial robot systems where traditional non-DRL-powered controllers are pervasive.

One interesting future direction is to incorporate more base controllers for one single task or to use a parameterized base controller distribution. Such a distribution might facilitate diverse control strategies and in turn aid the exploration of the agent. Another possible direction is to use Task and Motion Planning algorithms [47] to automatically synthesize base controllers for tasks requiring multistep planning.

REFERENCES

[1] W. He and Y. Dong, “Adaptive fuzzy neural network control for a constrained robot using impedance learning,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 4, pp. 1174–1186, Feb. 2018.

[2] L. Kong, W. He, C. Yang, Z. Li, and C. Sun, “Adaptive fuzzy control for coordinated multiple robots with constraint using impedance learning,” IEEE Trans. Cybern., vol. 49, no. 8, pp. 3052–3063, Aug. 2019.

[3] L. Kong, W. He, Y. Dong, L. Cheng, C. Yang, and Z. Li, “Asymmetric bounded control for an uncertain robot by state feedback and output feedback,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 51, no. 3, pp. 1735–1746, Mar. 2021.

[4] Y. Yang, W. Gao, H. Modares, and C.-Z. Xu, “Robust actor–critic learning for continuous-time nonlinear systems with unmodeled dynamics,” IEEE Trans. Fuzzy Syst., vol. 30, no. 6, pp. 2101–2112, Jun. 2022.

[5] D. Silver et al., “Mastering the game of go without human knowledge,” Nature, vol. 550, no. 7676, pp. 354–359, 2017.

[6] O. Vinyals et al., “Grandmaster level in starcraft ii using multi-agent reinforcement learning,” Nature, vol. 575, no. 7782, pp. 350–354, 2019.

[7] B. Kiymaz, C. K. Yanmaz, H. Modares, and F. E. Lewis, “Optimal and autonomous control using reinforcement learning: A survey,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[8] Y. Hu, W. Wang, H. Liu, and L. Liu, “Reinforcement learning tracking control for robotic manipulator with kernel-based dynamic model,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 9, pp. 3570–3578, Sep. 2020.

[9] H. Li, Z. Qichao, and D. Zhao, “Deep reinforcement learning-based automatic exploration for navigation in unknown environment,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 6, pp. 2064–2076, Jun. 2019.

[10] M. Andrychowicz et al., “Learning dexterous in-hand manipulation,” Int. J. Robot. Res., vol. 39, no. 1, pp. 3–20, 2020.

[11] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2017, pp. 3389–3396.

[12] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, “Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2018, pp. 4238–4245.

[13] D. Kalashnikov et al., “Scalable deep reinforcement learning for vision-based robotic manipulation,” in Proc. Conf. Robot Learn., 2018, pp. 651–673.

[14] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming exploration in reinforcement learning with demonstrations,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2018, pp. 6292–6299.

[15] Y. Zhu et al., “Reinforcement and imitation learning for diverse visuomotor skills,” in Proc. Robot., Sci. Syst. (RSS), Pittsburgh, PA, USA, Jun. 2018, doi: 10.15607/RSS.2018.XIV.009.

[16] L. Hermann, M. Argus, A. Eitel, A. Amiranshahvili, W. Burgard, and T. Brox, “Adaptive curriculum generation from demonstrations for sim-to-real visuomotor control,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2020, pp. 6498–6505.

[17] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, “Goal-conditioned imitation learning,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 15298–15309.

[18] G. Xiang and J. Su, “Task-oriented deep reinforcement learning for robotic skill acquisition and control,” IEEE Trans. Cybern., vol. 51, no. 2, pp. 1056–1069, Feb. 2021.

[19] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, “Multiple interactions made easy (MIME): Large scale demonstrations data for imitation,” in Proc. Conf. Robot Learn., 2018, pp. 906–915.

[20] T. Johannink et al., “Residual reinforcement learning for robot control,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 6023–6029.

[21] G. Schoetterl et al., “Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2020, pp. 5548–5555.

[22] D. A. Pomerleau, “ALVINN: An autonomous land vehicle in a neural network,” in Proc. Adv. Neural Inf. Process. Syst., 1989, pp. 305–313.

[23] M. Bojarski et al., “End to end learning for self-driving cars,” in Proc. NIPS, 2016, pp. 1–9.

[24] A. Giusti et al., “A machine learning approach to visual perception of forest trails for mobile robots,” IEEE Robot. Autom. Lett., vol. 1, no. 2, pp. 661–667, Jul. 2016.

[25] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in Proc. Int. Conf. Artif. Intell. Statist., 2011, pp. 627–635.

[26] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, “EnsembleDagger: A Bayesian approach to safe imitation learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Nov. 2019, pp. 5041–5048.

[27] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning,” in Proc. ICML, 2000, pp. 663–670.

[28] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 4565–4573.

[29] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529–533, Feb. 2015.
F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, “Towards vision-based deep reinforcement learning for robotic motion control,” in Proc. Australas. Conf. Robot. Automat., 2015, pp. 1–8.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in Proc. Int. Conf. Learn. Represent., 2016, pp. 1–14.

M. Vecerik, O. Sushkov, D. Barker, T. Rothorl, T. Hester, and J. Scholz, “A practical approach to insertion with variable socket position using deep reinforcement learning,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 754–760.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1561–1570.

M. Andrychowicz et al., “Hindsight experience replay,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5048–5058.

Z. Yang, H. Qu, M. Fu, W. Hu, and Y. Zhao, “A maximum divergence approach to optimal policy in deep reinforcement learning,” IEEE Trans. Cybern., early access, Sep. 3, 2021, doi: 10.1109/TCYB.2021.3104612.

X. Yang et al., “Hierarchical reinforcement learning with universal policies for multistep robotic manipulation,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 1, 2021, doi: 10.1109/TNNLS.2021.3059912.

G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, “Uncertainty-aware reinforcement learning for collision avoidance,” 2017, arXiv:1702.01182.

B. Lutjens, M. Everett, and J. P. How, “Safe reinforcement learning with model uncertainty estimates,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 8662–8668.

Y. Wu et al., “Uncertainty weighted actor-critic for offline reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 11319–11328.

M. Vecerik et al., “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards,” 2017, arXiv:1707.08817.

S. Levine and V. Kolotun, “Guided policy search,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1–9.

J. Luo et al., “Reinforcement learning on variable impedance controller for high-precision robotic assembly,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 3080–3087.

C. J. C. H. Watkins and P. Dayan, “Q-learning,” Mach. Learn., vol. 8, nos. 3–4, pp. 279–292, 1992.

S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1587–1596.

E. Coumans and Y. Bai. (2016). Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. [Online]. Available: http://pybullet.org

N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, “Incremental task and motion planning: A constraint-based approach,” in Proc. Robot., Sci. Syst. (RSS), Ann Arbor, MI, USA, Jun. 2016, doi: 10.15607/RSS.2016.XII.002.

Guangming Wang (Graduate Student Member, IEEE) received the B.S. degree from the Department of Automation, Central South University, Changsha, China, in 2018. He is currently pursuing the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai, China. His current research interests include robot learning and computer vision, in particular, deep reinforcement learning for robot control.

Minjian Xin received the bachelor’s degree from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in 2020. He is currently pursuing the master’s degree in computer science with the University of California at San Diego, La Jolla, CA, USA.

His latest research interests include deep reinforcement learning and robot control.

Jiao Tong University. His research interests include autonomous mobile robots, computer vision, and autonomous driving.

Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2004 and 2007, respectively.

He is currently a Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His current research interests include visual servoing, service robots, computer vision, and autonomous driving.

Minjian Xin received the bachelor’s degree from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in 2020. He is currently pursuing the master’s degree in computer science with the University of California at San Diego, La Jolla, CA, USA.

His latest research interests include deep reinforcement learning and robot control.

Wenhua Wu is currently pursuing the B.S. degree with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China.

His latest research interests include deep reinforcement learning and robot control.

Zhe Liu received the B.S. degree in automation from Tianjin University, Tianjin, China, in 2010, and the Ph.D. degree in control technology and control engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016. From 2017 to 2020, he was a Post-Doctoral Fellow with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. From 2021 to 2022, he was a Research Associate with the Department of Computer Science and Technology, University of Cambridge, Cambridge, U.K. He is currently a tenure-track Associate Professor with the MoE Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University. His research interests include autonomous mobile robots, multirobot cooperation, and autonomous driving systems.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.