D2SR: Transferring Dense Reward Function to Sparse by Network Resetting

Yongle Luo[a,b], Yuxin Wang[a,b], Kun Dong[a,b], Yu Liu[a,b], Zhiyong Sun[a,b], *Member, IEEE*, Qiang Zhang[a,b] and Bo Song[#,a,b,c], *Member, IEEE*

***Abstract*— In Reinforcement Learning (RL), most algorithms use a fixed reward function, and few studies discuss transferring the reward function during learning. Actually, different types of reward functions have different characteristics. In general, a shaped dense reward function has the advantage of quickly guiding the agent to a high-value state but has the disadvantage of being difficult to design a well-shaped function and susceptible to noise. The sparse reward has the advantages of being robust and consistent with the task, but less efficient in early exploration. Therefore, this paper proposes an algorithm called Dense2Sparse by Network Resetting (D2SR), which simultaneously satisfies the efficiency of dense reward functions and the robustness of sparse rewards. Specifically, the D2SR method can rescue the agent from being misled by sub-optimal dense rewards by network resetting parameters and transferring experience to sparse rewards, thereby achieving significant improvements in the direction of the global optimum. In this study, through a series of ablation experiments on challenging robot manipulation tasks, we find that D2SR can reduce the requirement of dense reward function design, which can also balance efficiency and performance in tasks with noisy rewards.**

I. Introduction

Deep Reinforcement Learning (RL) achieved remarkable advancements in various decision-making domains, such as Go [1], games [2], autonomous driving [3] and robotic manipulation tasks [4], [5], etc. Most RL algorithms require a reward function to provide feedback [6]. The performance of RL agent's learning largely depended on the reward function design. Generally speaking, most of these RL algorithms only use a fixed reward function during the entire learning cycle [7]–[9], while few articles discuss switching different reward functions during the learning process of the agent [10], [11]. This study focuses on the reward function switching problem of RL, specifically for standard manipulation tasks, and proposes a robust and effective general switching scheme: Dense to Sparse by network resetting (D2SR).

Reward functions can be broadly categorized into two types: dense and sparse. Dense reward functions usually provide specific feedback to the agent at each state [12], allowing it to distinguish which actions are better. However, designing a global optimal dense reward function is challenging, and dense rewards are more susceptible to noise signals, which can propagate and amplify through the Bellman equation [13]–[15]. On the other hand, sparse reward functions use a binary reward indicating task completion, with a penalty reward for failure and another positive reward for success [4], [16], [17]. This type of reward is less susceptible to noise and enables the agent to learn the globally optimal policy if sufficient successful samples are available [18]. However, an agent learning from scratch may face a paradox while learning from scratch: "the agent needs a good policy to complete the task, if without completing the task, it only receives samples with the same penalty reward and cannot learn from these poor data to get a good policy".

The advantages and disadvantages of these two types of reward functions are so prominent and complementary. The dense reward function has high efficiency in early exploration, but designing a global optimal dense reward function is difficult, and it is susceptible to noise. On the other hand, the sparse reward function has low efficiency in early exploration, but if valuable samples are provided, the agent guided by sparse rewards can in principle have higher consistency with the task and is more robust to noise.

Therefore, the proposed solution aims to incorporate the benefits of both reward functions while avoiding their drawbacks. The previous proposal, "Dense2Sparse" (D2S) algorithm [11], utilizes dense rewards during the early stage for noisy manipulation tasks to acquire valuable data and then transfers it to a sparse reward function for fine-tuning in subsequent learning. However, D2S is a suboptimal solution in the reward function transformation scheme for complex manipulation tasks, and its tuning effect is relatively limited compared with dense.

Recently, Nikishin et al. [19] suggested that deep RL algorithms are susceptible to primacy bias, a tendency to overfit early samples preventing the agent from improving its policy on subsequent experiences. This work provides an efficient mechanism to alleviate the primacy bias, i.e., resetting the network. By resetting the network's parameters, the agent can recover plasticity and erase any prior bias acquired from suboptimal data.

This work inspires us to reconsider the issue of D2S. Despite using sparse samples for tuning in later stages, the bias originating from the early stage training of the network with noisy dense samples hinders the effectiveness of network learning. Thus, we aim to utilize the above "hard reset" technique to alleviate the bias, with the expectation of

This work is supported in part by NSFC (61973294), Anhui Provincial Key R&D Program (2022i01020020), and the University Synergy Innovation Program of Anhui Province, China (GXXT-2021-030).

[a]Institute of Intelligent Machines, Hefei Institute of Physical Science, CAS, Hefei, 230031, China

[b]University of Science and Technology of China, Hefei 230026, China

[b]Jianghuai Frontier Technology Coordination and Innovation Center, Hefei, 230088, China

[#]Corresponding author, email: `songbo@iim.ac.cn`

effectively leveraging the benefits of both dense and sparse rewards.

Specifically, this work first uses a dense reward function that is not necessarily globally optimal to guide the agent to quickly explore. After collecting a certain amount of high-value samples, the reward function is transferred to a sparse one. In addition, the network parameters need to be reset, and the rewards of the collected samples need to be recomputed with the sparse reward function. All subsequent learning and exploration processes are conducted under the guidance of the sparse reward function. We refer to this approach as Dense2Sparse by network resetting (D2SR). Compared with the previous D2S scheme, D2SR only needs to reset the network parameters, and recompute the past experiences with the sparse reward function. Nevertheless, these modifications lead to a substantial improvement in performance.

This study first demonstrates that D2SR can eliminate dense bias and efficiently approach the global optimum, providing clear advantages over D2S in robotic manipulation tasks. A comprehensive evaluation of the impact of each component of D2SR was conducted through detailed ablation experiments. Experimental results of four diverse dense reward functions indicate that D2SR can reduce the requirement of dense reward function. The robustness of D2SR was further investigated in a task with noisy reward signals. Overall, D2SR paves a promising path toward using different reward functions.

II. Related Work

A few deep RL algorithms discuss the topic of switching different reward functions. The successor-based algorithms [10], [20] decompose the value function into a predictive representation and a reward function, allowing for quick adaptation to new reward functions through updates to the reward model. However, these studies did not anticipate any additional performance benefits from the new reward functions. The closest related work is the D2S algorithm [11], which aims to merge the advantages of both dense and sparse reward functions. However, none of these studies took into consideration the bias induced in the network by the previous reward function, which can impact the effectiveness of the new reward function.

Recently, studies have approached this issue from different perspectives, including capacity loss [21], loss of plasticity [22] and primacy bias [19]. These articles have noticed that deep RL loses some of its ability to learn new data after learning the previous data. To address this issue, the present paper employs the "hard reset" scheme outlined in [19] to eliminate the bias introduced by dense reward samples, so as to better adapt to the learning of the new sparse reward function. Finally, regarding the collection of sparse reward data, compared to demonstration-based methods [18], the high-value samples of D2SR are not provided by human demonstrators, but only need to design a less demanding dense reward function for the RL agent to collect by itself.

III. Background

This paper mainly focuses on the reward transformation of off-policy actor-critic structure deep RL algorithm for manipulation tasks. Therefore, this section introduces some background, including RL and the classic actor-critic algorithm.

A. Reinforcement Learning

RL consists of an agent interacting with the environment. The agent learns from the feedback of the environment to optimize policy based on maximizing the total expected reward [13]. Considering the standard RL formed as a Markov decision process (MDP), which is defined by a set of states $s \in S$, a set of actions $a \in A$, dynamics $p(s'|s,a)$, a reward function $R(s,a)$ and a discount factor $\gamma \in (0,1)$. At each timestep t, the agent performs an action a_t according to the current state s_t. Subsequently, the agent receives a reward signal $r_t = R(s_t,a_t)$ and the next state s_{t+1} sampled from the distribution $p(s'_{t+1}|s_t,a_t)$. The RL agent chooses an action a_t from a policy $\pi(s_t)$, which maps states to actions directly. The ultimate goal of RL is to learn an optimal policy π^* by maximizing the expected return $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where T is the time horizon.

B. Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) [7] is a classic off-policy model-free RL algorithm with promising performance in continuous control tasks. As an actor-critic framework, DDPG utilizes a critic network to guide the policy learning: an actor network approximating the policy π_θ and a critic network learning the action-value function Q_ϕ.

For stability reasons, a set of target networks, π'_θ and Q'_ϕ, are utilized, where the networks have the same structure but undergo delayed updating. The parameters ϕ of the critic network are learned by minimizing the Bellman error, and the loss function is defined as follows:

$$Loss_\phi = |Q_\phi(s_t,a_t) - y_t|^2, \tag{1}$$

where the target $y_t = r_t + \gamma Q'_\phi(s_{t+1},a_{t+1})$ and the action a_{t+1} are generated by $\pi'_\theta(s_{t+1})$.

Based on the Q function, the policy network parameters θ are trained using the gradient descent method with the loss function defined in (2):

$$Loss_\theta = -\mathbb{E}_{s_t} Q[s_t, \pi_\theta(s_t)]. \tag{2}$$

The DDPG algorithm maintains a replay buffer to store transitions (s_t,a_t,s_{t+1},r_t) that have been experienced previously. This allows the agent to update itself based on a set of uncorrelated transitions, which helps to stabilize the training process by breaking the temporal correlations in the updates.

The well-known off-policy RL algorithms soft actor-critic (SAC) [23] and twin delayed DDPG (TD3) [8] are advanced versions of DDPG. In this work, we use TD3 as the backbone algorithm.

IV. Method

In contrast to standard RL approaches that only employ a single reward function, the proposed method, D2SR, is presented in detail in this section, which enables the RL agent to make more effective use of multiple reward functions for learning.

Specifically, this work explores the use of two distinct reward functions: dense rewards and sparse rewards. Dense rewards are known to facilitate quick learning, but are susceptible to becoming trapped in local optima. Conversely, sparse rewards are slow in early learning but more likely to converge to global optima. To demonstrate how to leverage the benefits of both reward functions, this work uses the standard manipulation task as a case study, showcasing that agents can be trained with D2SR to achieve improved efficiency and performance.

The intuitive solution is to use a trivial dense reward function initially to guide the agent to collect a sufficient number of high-value samples, followed by a switch to a sparse reward function for performance tuning purposes, as in D2S [11]. However, this scheme ignores two implicit issues that arise from the early dense reward samples.

One is that the dense reward samples remain in the experience buffer, which could introduce biases during the later update process. It is not recommended to directly delete these samples, as this would result in a substantial reduction of high-value data in the experience buffer, leading to a drastic alteration of the network's gradient and eventually causing the policy to collapse [24]. Therefore, we propose to recompute the dense reward samples with a more robust sparse reward function. This kind of re-computation of rewards does not alter the dynamics of the environment, thus avoiding the introduction of any new bias.

The second issue is the persistent primacy bias of the network parameters after training the dense reward samples, which impairs the learning of subsequent sparse reward samples. At present, there is no regularization method [19] that can simultaneously maintain the network's policy performance and restore its plasticity as random initialization. To address this issue, D2SR employs the hard reset technique outlined in [19] to alleviate the primacy bias. Although this approach will cause brief performance collapses, the replay buffer can be seen as a non-parametric model of the world [25], allowing the agent to quickly recover from brief online interactions and normal off-policy policy updates. Furthermore, in the D2SR framework, the high-value sparse reward samples can drive performance drastically beyond the previous dense reward samples after mitigating the primacy bias. To some extent, this is a "retreat to advance" strategy.

Finally, the D2SR scheme can reset the network and recompute the rewards for collected samples to merge the benefits of the high efficiency of the dense reward and the performance of the sparse reward. As a result, D2SR reduces the need for designing complex expert dense reward functions.

V. Experimental setup

In this section, we first introduce the robotic simulation environments utilized in our experiments. Subsequently, we describe the reward functions employed in our study, including four unique dense reward functions tailored to the characteristics of the manipulation tasks, as well as a general sparse reward function.

A. Task setup

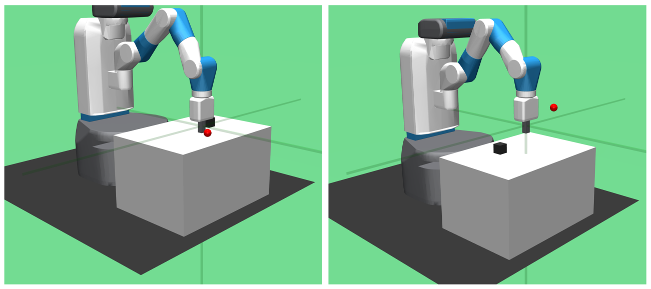

Fig. 1. Robotic simulation environment: FetchPush, FetchPickAndPlace.

The simulation environment utilized in our experiments is provided by OpenAI Gym [26], which offers a suite of complex continuous control tasks based on current robotic hardware. Throughout our experiments, we employ a 7-DOF Fetch robotic arm equipped with a two-fingered parallel gripper as our robotic agent. The simulation of the robot is carried out using the MuJoCo physics engine [27]. We evaluate our method D2SR in two challenging tasks, including FetchPush and FetchPickAndPlace, as illustrated in Fig. 1. In these tasks, the state representation includes the positions and linear velocities of the gripper and fingers, the positions, rotations, angular velocities of the object, and relative position and linear velocities with respect to the gripper. The action space is 4-dimensional, with 3 dimensions controlling the incremental movement of the gripper and the fourth dimension controlling the opening and closing of the fingers. As standard goal-conditioned tasks, the goal of these tasks is to reach a desired goal-object position, represented by the red dot in Fig. 1.

B. Reward functions

To demonstrate the compatibility of D2SR with various dense reward functions, in this section, we present four different dense rewards for the manipulation tasks under consideration. Subsequently, we introduce the general sparse reward function applied to these manipulation tasks.

In [4], it was shown that DDPG with HER can successfully learn a good policy for the standard manipulation tasks with sparse rewards by a goal-relabeling strategy, but not with a dense reward function. The dense reward function in HER, R_d^1, is expressed as follows:

$$R_d^1(s_t, a_t) = -d_2, \tag{3}$$

where d_2 represents the distance between the object and the goal of the object in meters (m).

As noted in [28], [29], it is crucial for the agent to be aware of the contact object for learning manipulation tasks. However, the original dense reward function (R_d^1) only takes into account the distance between the object and the goal of the object, which cannot effectively guide the agent to complete the task. To address this issue, we have designed three additional reward functions, which consider the additional item of the distance between the gripper and the object. The first of them is R_d^2.

$$R_d^2(s_t, a_t) = -d_1 - d_2, \tag{4}$$

where d_1 represents the distance between the gripper and the object in meters (m).

The second is R_d^3, which maps each item to $(0, 1)$.

$$R_d^3(s_t, a_t) = 1 - tanh(d_1) + 1 - tanh(d_2). \tag{5}$$

The anthers in [30] point out that positive rewards can negatively impact the exploration ability of a deep RL network. Therefore, to investigate the impact of reward shift on performance, the third dense reward function, R_d^4, was designed by adding an offset to R_d^2 to increase the absolute value of the reward.

$$R_d^4(s_t, a_t) = 1 - d_1 - d_2. \tag{6}$$

In this study, we leverage the standard sparse reward function, R_s, as used in the original HER algorithm for performance tuning. The sparse reward function is defined as follows:

$$R_s(s_t, a_t) = -\mathbb{I}(d_2 < 0.05), \tag{7}$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if its argument is true, and 0 otherwise. This reward function assigns a reward of -1 to the agent when the task is not completed and 0 when the task is completed. Task completion is defined as the condition where the distance between the object and the goal of the object, d_2, is less than 0.05 meters.

VI. Experimental Results and Analysis

This section provides a systematic assessment of the proposed D2SR method's performance compared to the baseline D2S on two standard manipulation tasks. Subsequently, we conduct a comprehensive ablation study to evaluate the effect of the new operation introduced in D2SR on performance. Furthermore, we demonstrate the compatibility of D2SR with various reward functions, thus reducing the expert knowledge requirement on dense reward function. Finally, we evaluate the robustness of D2SR in the presence of noisy reward signals.

A. Compared to Baselines

To evaluate the performance of the proposed method, we employ the backbone RL algorithm TD3 and the goal-relabeling technique introduced in the HER method to augment successful samples in goal-conditioned tasks. The evaluation comprises four methods, namely D2SR, D2S, Dense, and Sparse. Dense and Sparse use dense and sparse reward functions (R_d^3 and R_s, respectively) throughout the training process.

We apply the future strategy of HER with k = 4 to replace original desired goals with achieved goals in the trajectory during experience replay for all methods. We compare the mean test success rate of these four methods over five fixed random seeds, with the shadow area representing the standard deviation. The agent is trained for 300 epochs using one CPU core, with each epoch comprising 50 episodes in the FetchPush and FetchPickAndPlace environments. D2SR and D2S transfer the reward function from R_d^3 to R_s at 150 epoch. The hidden layer consists of 256 units, and the number of final output units is the total dimensions of the state and desired goal of the task. The parameters of network in TD3 are identical to those in DDPG+HER (OpenAI Baselines). We use a replay buffer with a size of 1e6 and train the policy with minibatch sizes of 2048.

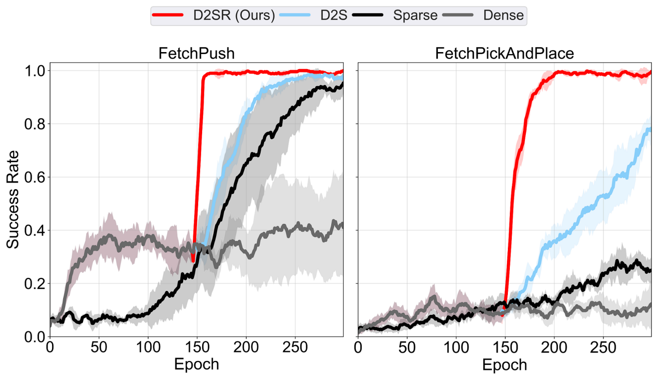

Fig. 2. Comparison on test success rate of D2SR and other methods in all two environments.

The learning curves are depicted in Fig. 2. The empirical results clearly indicate that both tasks suffer from suboptimal performance when using a singular fixed reward function. Specifically, the sparse reward function R_s encounters the exploration dilemma, particularly in the FetchPickAndPlace task, while the dense reward function R_d^3 leads the agent to sub-optimal performance, which are common phenomenon in RL. While the prior approach, D2S, shows a slight performance gain over Dense by introducing new sparse samples after 150 epochs, the improvement is not significant. In contrast, our proposed D2SR method achieves a significant improvement compared to Dense in both tasks, with a success rate of nearly 100% after only a few interactions. Despite the short-term performance decrease that D2SR experiences due to resetting network parameters, high-quality sparse reward samples retained in the buffer enable the agent to quickly recover and surpass the bottleneck of the previous dense rewards, leading to a substantial performance boost. Moreover, the curves of D2SR exhibit less shadow area after 150 epochs compared to the other three methods, indicating that the proposed approach is more stable and less sensitive to random seeds.

These results provide strong evidence that D2SR can simultaneously integrate the advantages of dense and sparse reward functions by network resetting and recomputing rewards, resulting in an impressive performance.

B. Ablation Studies

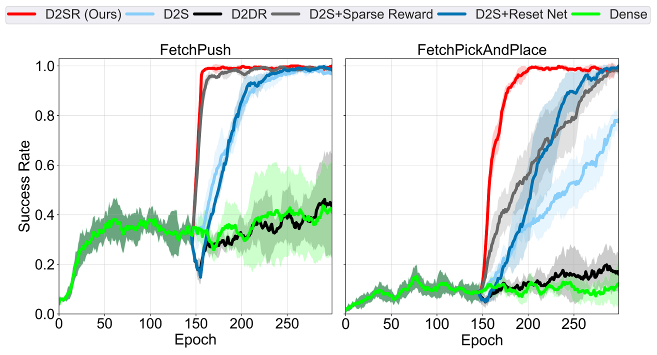

Fig. 3. Ablation results on two standard robotic environments.

In the aforementioned experiments, D2SR exhibited a substantial performance boost over D2S. To further investigate the contribution of each component in D2SR, we conducted ablation experiments. Fig. 3 illustrates the learning curves of D2S+Sparse Reward (D2SR without resetting the network). The results demonstrate that D2S+Sparse Reward performs significantly slower than D2SR, particularly in the FetchPickAndPlace task. Nevertheless, D2S+Sparse Reward still outperforms the previous method, D2S, in both tasks, thus confirming that samples in the buffer with dense rewards indeed hinder learning performance.

In contrast, the D2S+Reset Net method only resets the network and does not re-compute samples with R_s in the buffer. The experiments show that there is a significant performance decrease after 150 epochs in both tasks, and the recovery is slower compared to D2SR, yet the final performance is superior to D2S. These findings reveal that resetting the network enables the agent to improve its learning ability for new data. However, when there is interference with dense reward samples, the benefits of network resetting are limited. Therefore, by resetting the network and recomputing collected samples with a sparse reward function, D2SR not only recovers the learning ability of the network but also enhances the robustness of collected samples, ultimately leading to a significant performance improvement.

C. Different Dense Reward Functions

This subsection evaluates the performance of the D2SR algorithm with different dense reward functions, namely D2SR_d1 (R_d^1), D2SR_d2 (R_d^2), D2SR_d3 (R_d^3) and D2SR_d4 (R_d^4). The learning curves are depicted in Fig. 4. Four diverse dense reward functions have significant differences in the learning performance of each task in the first 150 epochs. The complete results indicate that the higher the performance achieved during learning with the original dense reward function, the more substantial improvement in the performance of D2SR. Nevertheless, even if the success rate is lower than 10% at 150 epoch, D2SR still demonstrates a considerable performance improvement relative to the original dense reward function. For those D2SR curves lower than the Sparse, we may obtain performance improvements by multiple resets in future work. In this study, we only transfer the reward function at a fixed epoch and take the success rate as a performance indicator. In future research, we may further find indicators to evaluate the quality of samples.

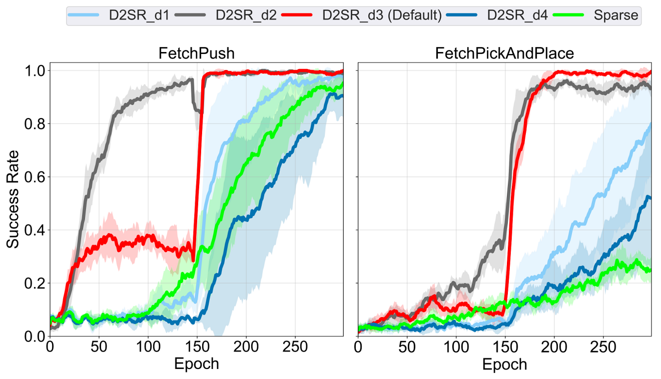

Fig. 4. Four different dense reward functions on two standard robotic environments.

In addition, there is an interesting discovery in this experiment. The original HER article claims that, in their attempts, even if the R_d^1 reward function is added with a linear item to encourage the gripper to approach the object, it cannot lead to successful training. Our experimental results demonstrate that different incentive items result in vastly divergent effects. Specifically, the dense reward function R_d^2 provides sufficient information to guide the agent and the agent was able to learn the FetchPush task within 150 epochs. In contrast, the dense reward function R_d^4 is unable to learn a good policy. We conjecture that an exploration perspective can offer an explanation: the positive reward signals will implicitly hinder exploration of the network [30]. To summarize, our experiments provide evidence that in the manipulation tasks, the HER algorithm can also effectively utilize dense reward functions, which are even more efficient than sparse rewards.

D. Robust to Noisy Rewards

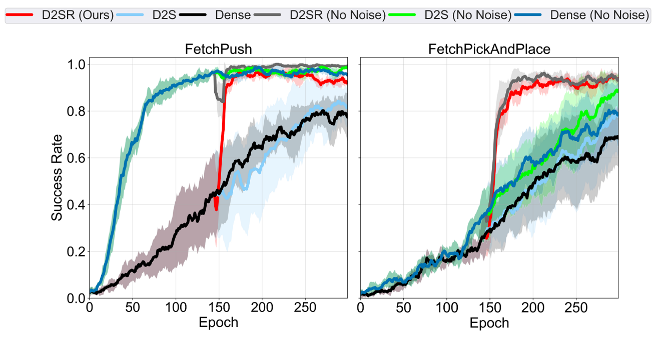

Fig. 5. The robustness of D2SR is evaluated on tasks that exist random noise in rewards and states.

In order to evaluate the robustness of D2SR under noisy rewards, we added noise to the states and rewards with a uniform distribution [-0.01, 0.01] in subsequent experiments. Based on the results presented in Section VI-C, we selected the dense reward function with the best performance, r_d^2, for further evaluation. The performance comparison of three methods with and without noise is shown in Fig. 5. We observe that the learning curve of Dense with noise has a significant drop in performance compared to Dense without noise (No Noise), especially in the FetchPush task. The previous method, D2S, does not show significant performance improvement in noisy environments. In contrast, D2SR is able to utilize the sparse reward function to significantly improve performance, and is minimally impacted by noise. These experiments demonstrate that D2SR can effectively leverage the robustness of the sparse reward function in the presence of noisy rewards.

VII. Discussion and Conclusion

In this work, we propose a concise and efficient reward function transformation method, D2SR, which integrates the advantages of both dense and sparse reward functions. Through a series of experiments on the manipulation tasks, we demonstrate that the proposed method D2SR can quickly break through the constraints of the original dense reward function and avoid the exploration dilemma when using a singular fixed sparse reward function, achieving both performance and efficiency. Moreover, the comparison experiments of four different dense reward functions reveal that D2SR can reduce the requirements for designing the dense reward function. Furthermore, the noise experiments demonstrate that D2SR can leverage the robustness of the sparse reward function, resulting in stable and improved performance for tasks with noisy rewards. The D2SR scheme enables the agent to use the dense reward function to collect high-quality samples and the sparse reward function for performance tuning, thereby paving a promising path for utilizing different reward functions. In future work, we will explore multiple transformations within the same and different reward functions to maximize sample efficiency.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] R. Zhang *et al.*, "Residual policy learning facilitates efficient model-free autonomous racing," *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[4] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[5] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[6] M. J. Mataric, "Reward functions for accelerated learning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 181–189.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] A. Zhang, H. Satija, and J. Pineau, "Decoupling dynamics and reward for transfer learning," *arXiv preprint arXiv:1804.10689*, 2018.

[11] K. Dong, Y. Luo, E. Cheng, Z. Sun, L. Zhao, Q. Zhang, C. Zhou, and B. Song, "Balance between efficient and effective learning: Dense2sparse reward shaping for robot manipulation with environment uncertainty," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2022, pp. 1192–1198.

[12] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[13] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, pp. 9–44, 1988.

[14] J. Wang, Y. Liu, and B. Li, "Reinforcement learning with perturbed rewards," *arXiv preprint arXiv:1810.01032*, 2018.

[15] Q. He and X. Hou, "Wd3: Taming the estimation bias in deep reinforcement learning," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2020, pp. 391–398.

[16] M. Plappert *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv:1802.09464*, 2018.

[17] B. Manela *et al.*, "Curriculum learning with hindsight experience replay for sequential object manipulation tasks," *Neural Networks*, vol. 145, pp. 260–270, 2022.

[18] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[19] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville, "The primacy bias in deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 828–16 847.

[20] E. Z. Liu, R. Keramati, S. Seshadri, K. Guu, P. Pasupat, E. Brunskill, and P. Liang, "Learning abstract models for strategic exploration and fast reward transfer," *arXiv preprint arXiv:2007.05896*, 2020.

[21] C. Lyle, M. Rowland, and W. Dabney, "Understanding and preventing capacity loss in reinforcement learning," *arXiv preprint arXiv:2204.09560*, 2022.

[22] S. Dohare, R. S. Sutton, and A. R. Mahmood, "Continual backprop: Stochastic gradient descent with persistent randomness," *arXiv preprint arXiv:2108.06325*, 2021.

[23] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[24] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Reincarnating reinforcement learning: Reusing prior computation to accelerate progress," *arXiv preprint arXiv:2206.01626*, 2022.

[25] H. P. Van Hasselt, M. Hessel, and J. Aslanides, "When to use parametric models in reinforcement learning?" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[26] G. Brockman *et al.*, "Openai gym," *arXiv:1606.01540*, 2016.

[27] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[28] B. Manela *et al.*, "Bias-reduced hindsight experience replay with virtual goal prioritization," *Neurocomputing*, vol. 451, pp. 305–315, 2021.

[29] Y. Luo, Y. Wang, K. Dong, Q. Zhang, E. Cheng, Z. Sun, and B. Song, "Relay hindsight experience replay: Continual reinforcement learning for robot manipulation tasks with sparse rewards," *arXiv preprint arXiv:2208.00843*, 2022.

[30] H. Sun, L. Han *et al.*, "Exploiting reward shifting in value-based deep rl," *arXiv preprint arXiv:2209.07288*, 2022.