A Study on Dense and Sparse (Visual) Rewards in Robot Policy Learning

Abdalkarim Mohtasib[1], Gerhard Neumann[2], and Heriberto Cuayáhuitl[1]

[1] Lincoln Centre for Autonomous Systems, University of Lincoln, Lincoln, UK
[2] Autonomous Learning Robots, Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract. Deep Reinforcement Learning (DRL) is a promising approach for teaching robots new behaviour. However, one of its main limitations is the need for carefully hand-coded reward signals by an expert. We argue that it is crucial to automate the reward learning process so that new skills can be taught to robots by their users. To address such automation, we consider task success classifiers using visual observations to estimate the rewards in terms of task success. In this work, we study the performance of multiple state-of-the-art deep reinforcement learning algorithms under different types of reward: Dense, Sparse, Visual Dense, and Visual Sparse rewards. Our experiments in various simulation tasks (Pendulum, Reacher, Pusher, and Fetch Reach) show that while DRL agents can learn successful behaviours using visual rewards when the goal targets are distinguishable, their performance may decrease if the task goal is not clearly visible. Our results also show that visual dense rewards are more successful than visual sparse rewards and that there is no single best algorithm for all tasks.

Keywords: Deep Reinforcement Learning · Reward Learning · Robot Learning

1 INTRODUCTION

In Deep Reinforcement Learning, the reward signal is typically carefully designed such that the agent can learn behaviour that achieves a good performance. But hand-coding and engineering rewards requires an expert to design it for each task to be learned, and it is often not easy to design rewards for robotic tasks. This limits the applications of DRL to real robots, especially when the end-user of the robot has to teach the robot new tasks. To address this limitation, it is crucial to find a mechanism that can autonomously and intuitively learn the rewards from a human expert for new tasks.

The problem of autonomous reward generation has been recently investigated in the literature by several researchers. Most previous works have used image-based success classifiers—as illustrated in Fig. 1—to learn the task's reward [12,13,21–23,25–27,30]. [21] attempted to use transfer learning to learn the rewards for new tasks, but with slow prediction times ($> 0.5s$ per interaction) that prevent its practical application. Other approaches used goal images to estimate the reward for each time step based on the difference between the goal and the current image, calculated in different ways [6,7,15, 16, 18, 19]. While these approaches achieved good results in learning the task reward, they have not investigated the effects of different types of rewards on DRL agents. There is no clear study that shows how the different DRL algorithms perform with different types of reward in different tasks.

The reward learning pipeline starts with collecting expert demonstrations for the task at hand. Their images are then labelled as success/no-success. Subsequently, the labelled data is used to train an image-based success classifier that estimates the success probability for each environment state. This success probability is used as a dense or sparse (visual) reward signal, see Section 3.2.

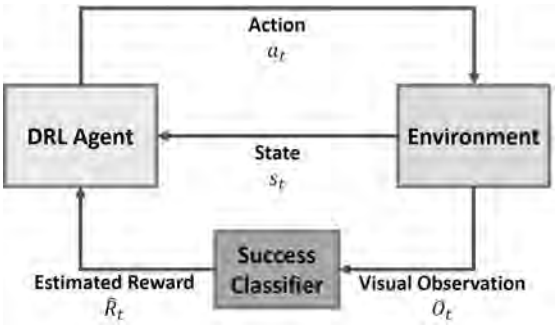


Fig. 1: System Overview.

The contribution of this paper is a comparison of different types of rewards (Dense, Sparse, Visual Dense, and Visual Sparse) for learning manipulation tasks. Our study was carried out using four different DRL algorithms (DDPG, TD3, SAC, and PPO) in four different robotic tasks. Our results show that it is indeed possible to learn good policies using visual rewards, where the higher the quality of the success classifier the better the learnt policy. Our results also show that, while a DRL algorithm may perform very well in one task, it may perform poorly in another.

2 Related Work

The literature shows different ways to learn numerical rewards. Some previous works have used Inverse RL to estimate the reward function from demonstrations [1, 3, 8, 9, 29]. Here, we consider a setting where the expert labels the visual observations as success/no-success. We use these expert labels to train a success classifier to estimate the reward. This setting differs from the Inverse RL setting (no expert labels). Other approaches use visual representations of the goal state to define the vision-based task [6, 7, 15, 16, 18, 19]. In these approaches, the goal image has been used in different ways to calculate the rewards: (i) using the latent distance between the current state image and the goal image [7, 15, 16]; (ii) using the pixel-wise L1 distance to the goal image [19]; or (iii) using the histogram distance to the goal image [6]. The approach of using a goal image to represent the task's success achieved good results. Yet this approach is limited as the goal could have different varieties and shapes. Furthermore, it is not always possible to represent the task goal using one or several images.

Our study focuses on the use of success classifiers to learn visual rewards of the task at hand [12, 13, 21–23, 25–27, 30]. The main neural network architecture that has been used for the task success classification is based on multiple convolutional blocks (convolutional layers followed by a max-pooling layer) followed by a multiple fully-connected layers [13, 22, 23, 26, 27, 30]. Sermanet et al. [21] used transfer learning of the Inception network [24] pre-trained for ImageNet classification [5] to extract the features from the environment's visual states. Subsequently, they used a simple neural network with multiple fully-connected layers to generate rewards from the extracted

visual features [21]. However, the interaction of such a large image classifier slows down the execution of the manipulation task.

While some of the previous works have used dense rewards in their experiments [13, 21, 23, 25, 27], some others have only employed sparse rewards [12, 22, 26, 30]. The difference between dense and sparse rewards is important because, in many tasks, the only available reward is a sparse reward and this represents a big challenge for the DRL agent to learn the task's objective. Furthermore, different types of RL algorithms have been used in these works such as DDPG [26], SAC [23], A3C [12, 22], REINFORCE [27], and DQN [25]. While task success classifiers have been used in different ways with different RL algorithms in the literature, there is no ablation study in the literature studying the pros and cons (or effects) of different types of rewards for inducing robot policies. This paper aims to fill that gap. Our ablation study, using different DRL algorithms across multiple tasks, reveals the effects of oracle dense rewards, oracle sparse rewards, visual dense rewards, and visual sparse rewards.

3 Research Methods

3.1 Problem Formulation

We consider environments that can be framed as a Markov Decision Process (MDP) [2], where an agent receives a reward r_t after taking action a_t in the state s_t, then it progresses to the next state s_{t+1}. We focus on the discounted case, where The agent tries to maximise the cumulative discounted reward $G_t = \mathbb{E}_\tau\left[\sum_{t=0}^{T}\gamma^t R_t\right] = \mathbb{E}_\tau\left[\sum_{t=0}^{T}\gamma^t r(s_t,a_t)\right]$, where γ is the discount factor, $\tau = (s_0, a_0, \cdots)$ denotes the whole trajectory, $s_0 \sim p_0(s_0)$, $a_t \sim \pi(a_t|s_t)$, and $s_{t+1} \sim p(s_{t+1}|s_t,a_t)$. We consider a success classifier $\hat{R}_t = f(o_t)$, where o_t is a visual observation of the environment (an image), and $\hat{R}_t \in [0,1]$ is the probability of having achieved the task in state s_t. We train $f(o_t)$ for a new manipulation task from N demonstrations by updating the parameters of this function to minimize $\sum \mathscr{L}(f(o_i), y_i)$, where $\mathscr{L}$ is the classification loss (cross entropy loss and mean square error in our case) and y_i is the image label. We assume that a demonstrator classifies the ground truth images, which are used by such a probabilistic classifier to learn to generate rewards. The research question that our study aims to answer is: *Can DRL agents learn good policies by using visual rewards derived from task success classifiers?*

3.2 Rewards

For each task, we trained DRL agents using four different types of rewards in order to understand the effects of the different types. The agents were trained using true Dense and Sparse rewards, where they come directly from the physical simulator. The equations of Dense and Sparse rewards are shown in Table 1. In addition, we used Visual Dense and Visual Sparse rewards, which were calculated based on the estimated success probability using our (best) CNN-based success classifiers. While the Visual Dense rewards for all tasks were estimated according to $\hat{R}_t = 2 \times P(success = 1|o_t) - 1$, the Visual Sparse rewards were estimated according to $\hat{R}_t = \begin{cases} 0, & P(success = 1|o_t) \geq 0.5 \\ -1, & P(success = 1|o_t) < 0.5 \end{cases}$

Where $P(success = 1|o_t)$ is the success probability estimated by the success classifier.

Table 1: Rewards for training DRL agents. ϕ is the tilt angle of the pendulum in *radians*, D_R is the distance between the end-effector of the robotic arm and target position in the Reacher task, D_P is the distance between the object and target location in the Pusher task, and D_F is the distance between the gripper of the Fetch arm and target position.

Reward	**Pendulum**	**Reacher**	**Pusher**	**Fetch**
Dense	$-\lvert\phi\rvert$	$-D_R$	$-D_P$	$-D_F$
Sparse	$\begin{cases} 0, & \lvert\phi\rvert < 0.15 \\ -1, & \lvert\phi\rvert \geq 0.15 \end{cases}$	$\begin{cases} 0, & D_R \geq 0.01m \\ -1, & D_R < 0.01m \end{cases}$	$\begin{cases} 0, & D_P \geq 0.01m \\ -1, & D_P < 0.01m \end{cases}$	$\begin{cases} 0, & D_F \geq 0.01m \\ -1, & D_F < 0.01m \end{cases}$

3.3 Task Success Classifiers

We compare two different image classifiers trained using expert demonstrations, and use them to reward the DRL agents. The image classifiers are as follows.

- **CNN Classifier (CNN)** This is a standard CNN-based model that has been used in literature [7, 9, 13, 21, 23, 25, 26, 30]. Its inputs are ($160 \times 160 \times 3$) resized images of the robotic environment, followed by six main convolutional blocks and one convolutional layer, see Fig. 2.
- **Time-Based CNN Classifier (T-CNN)** This architecture extends the CNN one with two pathways and features (shared in between): one is the classification path, the other is a timing path that predicts the proportion of task completion (a regressor), see Fig. 2. The task completion proportion for each image is calculated according to $y_t = \frac{t}{(j-1)}$, where t is a given time step, and j is the total number of time steps in the demonstration at hand. The timing path will add more gradient information and this aims to be helpful in predicting the task success.

3.4 Training Methodology

For each task, we collected a set of 10 successful demonstrations in different tasks (see Section 4.1). These demonstrations are used for training the success classifiers in each task. Each image in these demonstrations is labeled as success/no-success. We compare the performance of the classifiers across all tasks and use the best classifier to estimate the success probabilities from visual observations. Thereafter, we train DRL agents using four different learning algorithms[3] (DDPG [14], TD3 [10], SAC [11], and PPO [20]) with dense rewards and sparse rewards across four different tasks. Similarly, another group of DRL agents are trained but using visual dense rewards.

[3] We used a PyTorch implementation of the DRL algorithms [17].

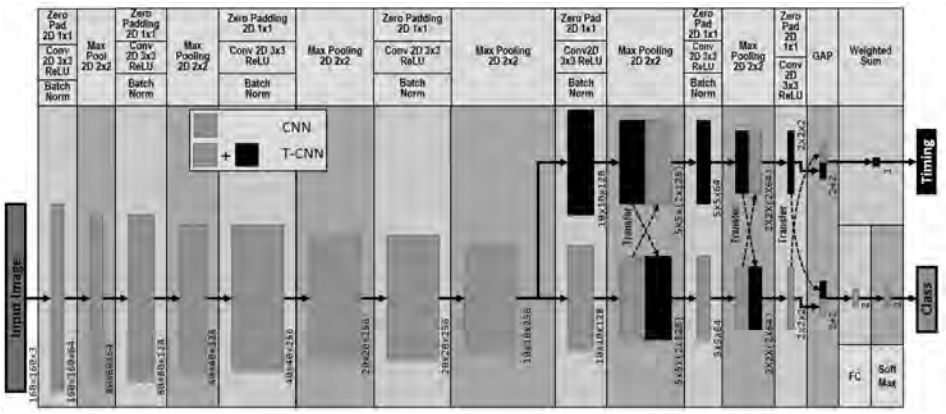


Fig. 2: Model architectures for task success classification with input images of (160 × 160 × 3). The Class output is the predicted success probability for both models. On the other hand, the Timing output is associated only with the T-CNN model. This output is the estimated task completion proportion (notation: **GAP**=Global Average Pooling).

4 Experiments and Results

4.1 Training Tasks

We trained the DRL agents using the following OpenAI Gym Environments [4], see Fig. 3: (1) **Pendulum**. A simple one Degree-Of-Freedom (DOF) task with one continuous action to stabilize the inverted pendulum in the up position. In each episode, the pendulum initial tilt angle is random. (2) **Reacher**: In this task, the end-effector (the green point, see Fig. 3) of the two links robotic arm (2-DOFs) should reach the red target. The position of the red target is initialised randomly in each episode. (3) **Pusher**: The 7-DOFs robotic arm in Fig. 3 pushes the white object to the red target position. The position of the white object is initialised randomly in each episode. (4) **Fetch (Reach)**: The 7-DOFs Fetch robotic arm in Fig. 3 should reach the red target position that is initialised randomly in each episode. This is a realistic robotic task that simulates the real Fetch robot (https://fetchrobotics.com).

Fig. 3: Visualisation of our simulation tasks: Pendulum, Reacher, Pusher, Fetch (Reach).

Table 2: Performance results of the CNN and T-CNN classifiers (notation: **ACC**=Average Classification Accuracy, **AUC**=Area Under the Curve).

	CNN					T-CNN				
Task	**ACC**	**Precision**	**Recall**	**F1 Score**	**AUC**	**ACC**	**Precision**	**Recall**	**F1 Score**	**AUC**
Pendulum	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Reacher	0.738	0.970	0.704	0.816	0.962	0.872	0.970	0.872	0.918	0.982
Pusher	0.990	0.992	0.994	0.993	1.000	0.992	0.994	0.994	0.994	1.000
Fetch	0.898	0.908	0.982	0.943	0.976	0.948	0.966	0.975	0.970	0.989
Average	0.907	0.968	0.920	0.938	0.985	**0.953**	**0.990**	**0.960**	**0.971**	**0.993**

4.2 Success Classifiers Results

The CNN (Standard Convolutional Neural Net) and T-CNN (Time-Based CNN) image classifiers in each of the four tasks were trained with a set of 10 demonstration episodes and tested with another set of 10 demonstration episodes. Here, we test the ability of the success classifier to predict the success probability for each observation (image) in the test set. We assess the performance of success classification according to the following metrics: Classification Accuracy, Precision, Recall, F1-score, Area Under the Curve. Table 2 shows the test results of our classifiers, where the T-CNN classifier outperformed the CNN classifier in all classification metrics across all tasks. This suggests that the additional gradient information for predicting the task completion proportion helps in predicting the task success. Thus, the T-CNN model is adopted to estimate the success probabilities for the visual rewards.

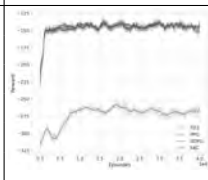
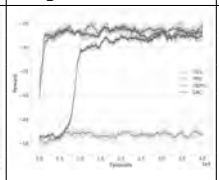
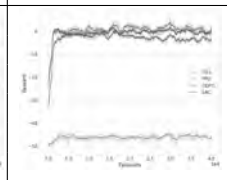
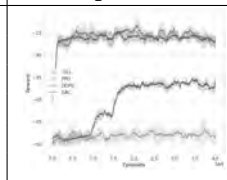
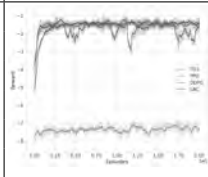
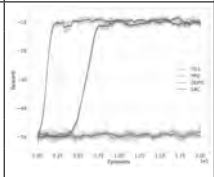
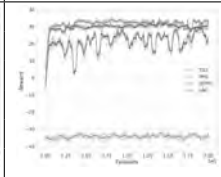
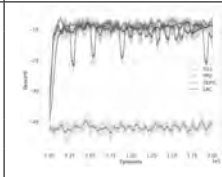
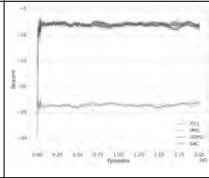
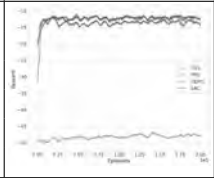
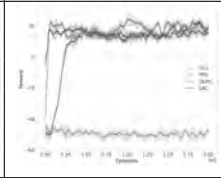
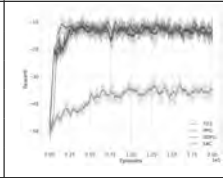
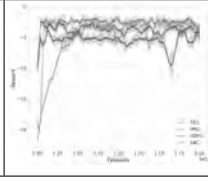
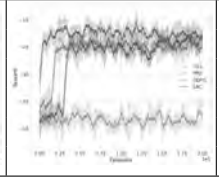
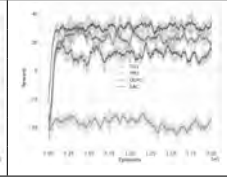
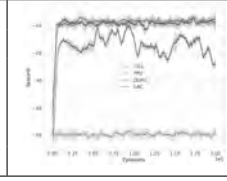
4.3 Experimental Results of the DRL Agents

We evaluated different aspects in the performance of our DRL agents. First, we start with the learning curves of the DRL agents under the different settings as shown in Table 3. The most important outcome from these learning curves is that the DRL agents (except for PPO agents) were able to learn good policies by using only the visual rewards that come from the success classifier. See the following video for example behaviours of the trained DRL agents[4].

It is crucial to test the learned policies to ensure that the visual rewards can be used to learn useful behaviours that lead to the successful execution of the tasks. Table 4 shows the test results of the learned policies. It is clear from these results that the visual rewards are indeed helpful in learning good successful behaviours—as noted by their success rates across tasks. On average, there is a small drop in performance when using visual rewards as the success classifier is error-prone. It can be noted that the drop in performance when using visual rewards is larger in the Reacher and the Fetch (Reach) tasks. With further investigation and experiments, we found that when the target object is behind the robotic arm, the visual images are not reflecting the correct environment's

[4] Video: https://youtu.be/8zOqEQDBleU

Table 3: Learning curves of DRL agents using different learning algorithms (DDPG, TD3, SAC, PPO) across four tasks when trained using dense, sparse, visual dense, and visual sparse rewards. The agents used five different seeds, 320 learning curves in total.

	Dense Reward	**Sparse Reward**	**Visual Dense Reward**	**Visual Sparse Reward**
Pendulum				
Reacher				
Pusher				
Fetch (Reach)				

state. Thus, the success classifier fails to predict the correct success probability, and hence the drop in the performance in this task. The ranking of algorithms according to average success rate is as follows: DDPG (81.4%±18%), SAC (77.8%±20%), TD3 (75.7%±26%), and PPO (12.1%±21%).

A statistical analysis using the Wilcoxon Signed-Rank Test (paired) [28] on the results of Table 4 revealed the following. Comparing Dense Success Vs. Sparse Success, the p-values are: $p = 1e-4$ including PPO, and $p = 6e-7$ excluding PPO. Comparing Visual Dense Success Vs. Visual Sparse Success, the p-values are: $p = 4e-4$ including PPO, and p=0.016 excluding PPO. Whilst the first comparison supports our claim that dense rewards are better than sparse ones, the second supports the claim that visual dense rewards are better than visual sparse rewards.

We carried out another statistical analysis using the Wilcoxon Signed-Rank Test (paired) [28] to compare the ranking of algorithms according to average success rate. While comparing DDPG Vs. TD3 gives a p-value of 0.236, comparing DDPG Vs. SAC gives p =0.182. The differences are not significant and more comparisons are needed.

Table 4: Performance of DRL algorithms across tasks. The maximum episode's length is 100 steps. Training and test times of DRL agents: training in HH:MM, and test in seconds. While 2 million steps were used in the **Pendulum** task, 10 million steps were used in the other tasks. The learnt policies were tested with 1000 episodes in each task.

Task	Agent	Dense Reward		Sparse Reward		Visual Dense Rew.		Visual Sparse Rew.	
		Success Rate	Average Eps. Len.	Success Rate	Average Eps. Len.	Success Rate	Average Eps. Len.	Success Rate	Average Eps. Len.
Pendulum	**DDPG**	98%	62.36	96%	59.68	96%	63.18	87%	53.88
	TD3	100%	62.53	87%	60.56	99%	58.19	87%	63.9
	SAC	95%	66.37	75%	61.76	99%	62.7	38%	61.12
	PPO	1%	60.77	2%	61.19	2%	51.86	4%	61.66
Reacher	**DDPG**	100%	51.87	97%	53.79	45%	82.16	62%	75.35
	TD3	98%	54.64	8%	97.28	52%	78.34	33%	83.99
	SAC	98%	52.93	97%	56.09	45%	80.38	47%	73.95
	PPO	6%	98.13	8%	95.01	11%	95.12	10%	95.6
Pusher	**DDPG**	91%	60.44	90%	59.94	80%	60.52	72%	69.38
	TD3	92%	62.8	87%	67.95	83%	66.25	80%	69.89
	SAC	95%	58.17	90%	63.16	87%	67.21	84%	63.1
	PPO	2%	99.46	19%	91.78	4%	98.78	14%	95.27
Fetch	**DDPG**	91%	52.23	88%	60.95	58%	73.25	51%	74.78
	TD3	94%	53.87	82%	58.07	64%	67.53	65%	68.13
	SAC	81%	58.52	82%	62.5	72%	67.83	59%	75.29
	PPO	90%	52.75	4%	98.11	13%	92.95	4%	97.41
Avg. Success		77.00%	62.99	63.25%	69.24	56.88%	72.89	49.81%	73.92
Std. Success		37.02%	14.67	38.89%	16.03	34.28%	13.85	29.78%	13.13

Table 5 reports training and test times of various experimented settings. Considering the training time[5] of agents using visual rewards, their training time is almost twice than non-visual rewards. Although such long training times should be addressed in future work, this cost comes with a large benefit where there is no need to hand-code the reward functions. Similarly and in contrast to agents using non-visual rewards, the test times of agents using visual rewards increase by about 30ms for every environment step. We calculated the average test time for one environment step across tasks, which resulted in $\sim$45ms—acceptable for real robotic tasks.

Furthermore, we investigated the effects of the choice of success classifier by comparing policies using our baseline and proposed success classifiers—CNN and T-CNN, respectively. Results show that while the performance of agents is similar in simple tasks (74% of task success on avg. across algorithms for both classifiers in the Pendulum task), T-CNN-based agents outperform CNN-based agents in more complex tasks (61.7% and 47.8% of an overall average task success across tasks, respectively), suggesting that the higher the performance of success classifiers the better learnt policies.

[5] PC: **CPU**: Intel i7-6950 @ 3.00GHz, 10 cores. **RAM**: 32GB. **GPU**: NVIDIA TITAN X 12GB.

Table 5: Training and test times of DRL agents: training in HH:MM, and test in seconds. While 2 million steps were used in the **Pendulum** task, 10 million steps were used in the other tasks. The learnt policies were tested with 1000 episodes in each task.

	Task	Dense or Sparse Reward				Visual Reward			
		DDPG	**TD3**	**SAC**	**PPO**	**DDPG**	**TD3**	**SAC**	**PPO**
Training	Pendulum	05:45	05:47	09:10	04:12	16:22	16:15	22:27	10:30
	Reacher	27:02	19:18	45:50	12:16	45:24	51:42	77:56	18:40
	Pusher	29:07	27:49	35:01	12:11	47:00	45:59	60:35	29:58
	Fetch (Reach)	29:30	24:33	34:50	14:59	50:20	40:25	62:55	26:34
	Avg.	22:51	19:21	31:12	10:54	39:46	38:35	55:58	21:25
Test	Pendulum	17.03	17.28	16.22	16.75	46.02	33.15	33.90	43.35
	Reacher	16.53	17.42	17.75	17.18	45.60	43.55	47.63	49.03
	Pusher	17.54	17.09	15.84	16.76	44.53	45.17	47.93	46.59
	Fetch (Reach)	16.14	17.56	17.38	17.78	45.72	45.34	45.87	47.13
	Avg.	16.81	17.34	16.80	17.12	45.47	41.80	43.83	46.53

5 Conclusion and Future Work

This paper shows that it is indeed possible to learn successful policies from visual rewards, though with higher computational cost than non-visual rewards. Our experiments reveal the following. First, dense rewards can achieve higher task success than sparse rewards. Second, the better the success classifier the better the policy. Third, when images do not represent the correct state of the environment, this may lead to learning poor policies. Fourth, while one algorithm might be good in a given task, it may not performs well in another task. DDPG achieved the highest task success across tasks, but the differences in performance against other algorithms were not significant.

Future work will consist of investigating the proposed learned visual rewards on real robotic tasks and multiple robot platforms. Other future works with high potential contribution to the previous work include accelerating the training times of DRL agents, and improving their success rates across tasks.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: ICML (2004)
2. Bellman, R.: A markovian decision process. Journal of mathematics and mechanics **6**(5) (1957)
3. Boularias, A., Kober, J., Peters, J.: Relative entropy inverse reinforcement learning. In: AISTATS (2011)
4. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
6. Edwards, A., Isbell, C., Takanishi, A.: Perceptual reward functions. arXiv preprint arXiv:1608.03824 (2016)

7. Edwards, A.D., Sood, S., Isbell Jr, C.L.: Cross-domain perceptual reward functions. arXiv preprint arXiv:1705.09045 (2017)
8. Finn, C., Levine, S., Abbeel, P.: Guided cost learning: Deep inverse optimal control via policy optimization. In: ICML (2016)
9. Fu, J., Singh, A., Ghosh, D., Yang, L., Levine, S.: Variational inverse control with events: A general framework for data-driven reward definition. NIPS **31** (2018)
10. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: ICML (2018)
11. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: ICML (2018)
12. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397 (2016)
13. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. IJRR **37**(4-5) (2018)
14. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: ICLR (2016)
15. Nair, A., Bahl, S., Khazatsky, A., Pong, V., Berseth, G., Levine, S.: Contextual imagined goals for self-supervised robotic learning. In: CoRL (2020)
16. Nair, A.V., Pong, V., Dalal, M., Bahl, S., Lin, S., Levine, S.: Visual reinforcement learning with imagined goals. NIPS **31** (2018)
17. Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., Dormann, N.: Stable baselines3. https://github.com/DLR-RM/stable-baselines3 (2019)
18. Sampedro, C., Rodriguez-Ramos, A., Gil, I., Mejias, L., Campoy, P.: Image-based visual servoing controller for multirotor aerial robots using deep reinforcement learning. In: IROS (2018)
19. Schoettler, G., Nair, A., Luo, J., Bahl, S., Ojea, J.A., Solowjow, E., Levine, S.: Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. arXiv preprint arXiv:1906.05841 (2019)
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
21. Sermanet, P., Xu, K., Levine, S.: Unsupervised perceptual rewards for imitation learning. arXiv preprint arXiv:1612.06699 (2016)
22. Shelhamer, E., Mahmoudieh, P., Argus, M., Darrell, T.: Loss is its own reward: Self-supervision for reinforcement learning. In: ICLR (2017)
23. Singh, A., Yang, L., Hartikainen, K., Finn, C., Levine, S.: End-to-end robotic reinforcement learning without reward engineering. In: RSS (2019)
24. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
25. Tung, H.Y., Harley, A., Huang, L.K., Fragkiadaki, K.: Reward learning from narrated demonstrations. In: CVPR (2018)
26. Vecerik, M., Sushkov, O., Barker, D., Rothörl, T., Hester, T., Scholz, J.: A practical approach to insertion with variable socket position using deep reinforcement learning. In: ICRA (2019)
27. Wang, X., Chen, W., Wang, Y.F., Wang, W.Y.: No metrics are perfect: Adversarial reward learning for visual storytelling. In: ACL (2018)
28. Wilcoxon, F.: Individual comparisons by ranking methods. In: Breakthroughs in statistics, pp. 196–202. Springer (1992)
29. Wulfmeier, M., Wang, D.Z., Posner, I.: Watch this: Scalable cost-function learning for path planning in urban environments. In: IROS (2016)
30. Xie, A., Singh, A., Levine, S., Finn, C.: Few-shot goal inference for visuomotor learning and planning. In: CoRL (2018)