Review of Deep Reinforcement Learning for Robot Manipulation

Hai Nguyen, Hung Manh La

***Abstract*—Reinforcement learning combined with neural networks has recently led to a wide range of successes in learning policies in different domains. For robot manipulation, reinforcement learning algorithms bring the hope for machines to have the human-like abilities by directly learning dexterous manipulation from raw pixels. In this review paper, we address the current status of reinforcement learning algorithms used in the field. We also cover essential theoretical background and main issues with current algorithms, which are limiting their applications of reinforcement learning algorithms in solving practical problems in robotics. We also share our thoughts on a number of future directions for reinforcement learning research.**

I. Introduction and Review Methodology

Reinforcement learning (RL) has attracted a lot of attention [1]–[3] in recent years with exciting results in multiple domains such as surpassing human experts on Atari games [4] and the game of Go [5]. Within the robotic manipulation context, RL offers a framework and a set of tools for learning dexterous manipulation end-to-end, directly from raw pixels. Initial successes in the field were promising, however, they revealed some inherent difficulties for applying RL to solve practical robotic challenges. Our review aims to provide our perceived picture of using RL in the context of robot manipulation. We try to cover background knowledge, interesting research results, open problems, and provide our insights into future directions.

We decided to conduct the review starting from the year 2013. We chose this year because most of the previous RL algorithms used in robotics before this year have been covered thoroughly in [6]. To our knowledge, from this year, there is no significant review in RL in the context of robot manipulation regardless of a number of interesting results. We try to bridge the gap by this review with the focus on the applications of RL for robotic manipulators. We conducted an extensive literature search in well-known electronic databases including IEEE Xplore, Google Scholar, and arVix. Articles will be included when they met the following criteria: a) They were written in English; b) They were published from 2013 onwards; c) The journal publications would be included only when there were significant differences with the conference paper; d) The content is relevant to deep RL for robot manipulation. The keywords were used for searching are *reinforcement learning*, *deep reinforcement learning*, and *robot manipulation*. We used these keywords and their combinations to filter out 42 papers with relevant content.

Hai Nguyen and Dr. Hung La are with the Advanced Robotics and Automation (ARA) Laboratory, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA. Corresponding author: Hung La, email: hla@unr.edu

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) Grant No. NNX15AI02H issued through the NVSGC-RI program under sub-award No. 19-21, and the RID program under sub-award No. 19-29, and the NVSGC-CD program under sub-award No. 18-54.

Our paper is organized as followed: we start in section II by describing key RL concepts at and terminology. Next, section III continues with a taxonomy of RL algorithms followed by section IV with a focus on the context of robot manipulation. We describe our perspectives about future directions in section V and section VI concludes the review.

II. Key Concepts and Terminology

Considering the standard RL setting when an agent interacts with an environment as shown in Figure 1. Beyond the agent and the environment, there are four main elements of a RL system: a policy, a reward signal, a value function, and optionally a model of the environment.

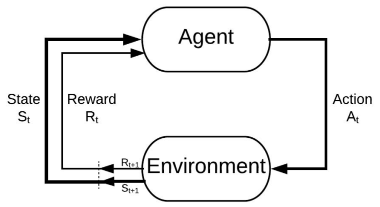

Fig. 1: The agent-environment interaction (credit [7]).

A *state* s is a complete description of the state of the world. There is no information about the world which is hidden from the state. A *policy* defines the learning agent's way of behaving at a given time. It is a mapping from the perceived states of the environment to actions to be taken when being in those states. Policy can be stochastic giving a distribution of actions over the current state $a = \pi(.|s)$ or can be deterministic $a = \mu(s)$.

A *reward signal* defines the goal of a RL problem. On each time step, the environment sends to the agent a single scalar number called the reward. A reward is dependent on the current state and the action just taken $r = R(s, a)$. The agent's ultimate goal is to maximize the cumulative reward that it receives over a long run. In general, we seek to maximize the expected return G_t with discount rate $\gamma \in [0\ 1]$:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

A *model* of the environment allows inferences to be made about how the environment will behave. The dynamics of the environment is fully characterized by a distribution p, which can also be deterministic $s' = p(s, a)$ or stochastic $s' = p(.|s, a)$.

Markov Decision Process (MDP) is a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. Thus MDPs involve delayed reward and the need to trade-off immediate and delayed reward. It contains:

- A set of possible states $\mathcal{S}$
- A set of possible actions $\mathcal{A}$
- A reward function $R(s, a) \in \mathcal{R}$
- A probability distribution $p(s', r|s, a)$ of the environment

A *value function* specifies what is good in the long run of a state s or a state-action pair (s, a) when following a particular policy π. We have the value function $V^{\pi}(s)$ of a state s or the action-value function $Q^{\pi}(s, a)$ under a policy π.

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi}[R(\tau)|s_t = s]$$
$$Q^{\pi}(s, a) = \mathbb{E}_{a \sim \pi}[R(\tau)|s_t = s, a_t = a]$$

The optimal value function is the value function with π being the optimal policy.

$$V^{*}(s) = \max_{\pi} \mathbb{E}_{a \sim \pi}[R(\tau)|s_0 = s]$$
$$Q^{*}(s, a) = \max_{\pi} \mathbb{E}_{a \sim \pi}[R(\tau)|s_0 = s, a_0 = a]$$

Given the optimal $Q^{*}(s, a)$, we can obtain the optimal action at a given state s, and then we can directly construct the optimal policy π^{*}:

$$a^{*}(s) = arg\ \max_{a} Q^{*}(s, a)$$

Bellman equation is obeyed by all four types of value functions $V^{\pi}(s)$, $Q^{\pi}(s, a)$, $V^{*}(s)$ and $Q^{*}(s, a)$:

$$V^{\pi}(s) = \mathbb{E}_{s' \sim \pi}[r(s, a) + \gamma V^{\pi}(s')]$$
$$Q^{\pi}(s, a) = \mathbb{E}_{s' \sim \pi}\left[r + \mathop{\mathbb{E}}_{a' \sim P}[Q(s', a')]\right]$$
$$V^{*}(s) = \mathbb{E}[r(s, a) + \gamma V^{*}(s')]$$
$$Q^{*}(s, a) = \mathbb{E}_{s' \sim \pi^{*}}[r + \gamma \max_{a'} Q(s', a')]$$

III. Taxonomy of RL Algorithms

It is not easy to come up with a distinct all-compassing taxonomy of all RL algorithms used in robot manipulation. We will cover only the main branches and the trade-offs that go with each class. We chose to include model-based vs. model-free and policy-based vs. value-based algorithms. Figure 2 from OpenAI's Spinning Up[1] showed a more complete list of algorithms.

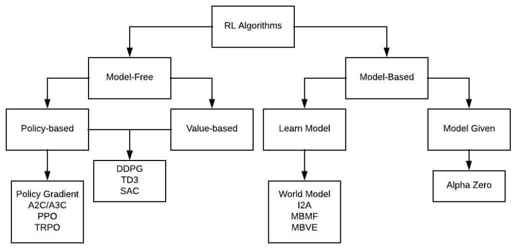

Fig. 2: Taxonomy of RL algorithms.

[1]https://spinningup.openai.com/en/latest/spinningup/rlintro2.html

A. Model-Based and Model-Free

We can divide available RL algorithms by deciding whether the agent has the access or learns a model of the environment. Having a model in hands allows the agent to plan ahead to predict state transitions and future rewards. This ability allows the agent to predict what would happen when choosing a certain action from a range of possible ones. If the model is correct then the learning would be greatly benefited in terms of sample efficiency compared to model-free methods. However, it is usually the case when the model is not available or the learning of such a model is very challenging. One challenge is when learning the model, the agent might introduce bias and as a result, the agent might perform extremely well with the learned model but might behave sub-optimally or even poorly with the real environment. Model-free methods focus on figuring out the value functions directly from the interactions with the environment. Algorithms in this class rely heavily on reward signals for learning the value functions. Hence, it is important to have learning-induced reward functions. Moreover, they are often easier to implement and tune hyperparameters. Currently, due to these advantages, model-free methods are used more often than their model-based counterparts.

B. Policy-Based and Value-Based

Value-based algorithms try to estimate the action-value function $Q(s, a|\theta)$ for the optimal $Q^{*}(s, a)$. This optimization is often performed off-policy. It means that the policy used to generate behaviour for getting training data, may be unrelated to the policy that is evaluated and improved, called the estimation policy. The optimum policy is retrieved by:

$$a = arg \max_{a} Q(s, a).$$

Policy-based methods parameterize the policy as $\pi(s, a|\theta)$, and the target is to optimize θ either through gradient descent on an objective function $J(\pi)$ or by maximizing local approximations of J. Policy-based methods are often on-policy, meaning that they estimate the value of a policy while using it for control. As a result, they are less sample-efficient as they only use samples collected from the latest version of the policy.

When using policy-based methods, we directly optimize what we need. This allows stability and reliability improvements. Q-learning methods, on the other hand, are indirect by estimating Q based on an objective function. There are many factors that can fail the learning of this kind, therefore, these methods are less stable. However, the main advantage of these algorithms is significantly more sample-efficient because they reuse data more effectively. This, in particular, is important when implementing on real robots.

IV. RL for Robot Manipulation

RL in the context of robotics, in general, is often represented with continuous high-dimensional action and state space. For robot manipulation, collecting samples is often expensive and time-consuming. Experiences are also sensitive to a variety of noise and difficult to reproduce. To collect a single training sample, it might take a few minutes for a robot to move around or perform the tasks. Robotics RL is often modelled as partially observable MDP as it is common for states to be unobservable or partially observable. Successful algorithms especially model-based methods, therefore, need to be robust to a significant scale of uncertainty in the model. In this section, we discuss three main issues, from our perspective, are limiting the application of RL for real-world robotics problems. We include sample inefficiency, exploration & exploitation, and generalization & reproducibility.

A. *Sample Inefficiency*

Sample inefficiency is one of the main reasons that seriously limit the applications of RL in robot manipulation. Even some of the best current RL algorithms can still be impractical due to sample inefficiency. There are multiple causes for the problem. Firstly, many algorithms try to learn to perform a task from scratch, therefore, they would need a lot of data to learn. Secondly, algorithms are still not good enough at exploiting useful information from current data. Some on-policy algorithms even require new data for every update step. Finally, data collection in robotics is often very time-consuming.

1) Brief Review: Evolution algorithms are the least sample-efficient as they do not use gradient for optimization, but they might have comparable performance. Evolution strategies used in [8] were able to match the performance in Atari games from [9] with 3-10x more data. The asynchronous version of actor-critic A3C [10] is more data-efficient, being able to surpass [9] on the Atari domain just by training on a multi-core CPU instead of a GPU. Policy gradient methods such as [11] is next in terms of sample-efficiency, followed by methods that uses replay buffer to estimate Q-values such as Deep Deterministic Policy Gradients (DDPG) [12] and Normalized Advantage Functions (NAF) [13]. Model-based algorithms are taking the lead in terms of data efficiency as they try to derive a model of the environment and use that model for training the policy instead of data from real interactions. Guided Policy Search [14] is very data-efficient as it uses trajectory optimization to direct policy learning and avoid poor local optima. The current winner is model-based "shallow" algorithm such as Probabilistic Inference for Learning COntrol (PILCO) [15]. [16] used PILCO and only needed about 4 minutes to learn a complex task such as the block-stacking task, and the time was reduced to 90 seconds when using knowledge transfer.

2) Open Problems: In order to be more data-efficient, we need to collect more data and use the data that we currently have more efficiently. One way to have more data is using multiple robots to collect data simultaneously as shown in Figure 3. Real data can also be augmented with synthetic data, possibly from simulators, and this approach has been adopted in a number of research [17]–[19]. In this approach, the gap between synthetic data and real robot data needs to be reduced so that the simulated data can be useful. The gap is quantified in [20] in a grasping task so that the difference will also be minimized during the learning process. [21] used deep learning architecture to map from synthetic images to real images. To bridge the reality gap, [18] used *progressive networks* for reusing from low-level visual features to high-level features in new tasks using transfer learning. We also need a mechanism to share data as in supervised domain with many useful public datasets. In robotics, however, data is specific to certain robots and configurations. It will be very useful if we have a mechanism to transform data so that it can be widely distributed and used in multiple platforms and configurations. Finally, we will need novel algorithms that can use data more efficiently. Model-based approach might be one of the most potential ways to unlock sample efficiency.

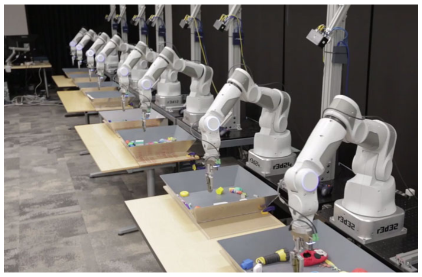
Fig. 3: Multiple robots to collect data (credit [22]).

B. *Exploration and Exploitation*

As an RL agent need to constantly take actions based on the current state and action, the fundamental question of whether to *explore* or *exploit* happens every time it acts. While exploration gives more knowledge about the environment, which might lead to better future decisions, exploitation chooses the best action to take given the current information that we have, narrowing us down to the current most promising direction. The best strategy will involve sacrificing short-term rewards for more reward in the future, meaning that a balance between exploration and exploitation is required.

1) Brief Review: Deep Q-Network (DQN) [4] used ϵ-greedy [23] to balance exploration and exploitation. With this strategy, the agent will either take random action at the probability ϵ or follow the action that maximizes the Q-value with the probability 1 - ϵ. Other variants existed such as decaying ϵ-greedy reducing ϵ over time and adaptive version [24] with ϵ being adjusted based on the temporal-difference. Vanilla policy gradient method, Trust Region Policy Optimization (TRPO) [25], and Proximal Policy Optimization (PPO) [26] explore by sampling actions according to the latest version of its stochastic policy. DDPG [12] trains a deterministic policy in an off-policy way with noise being added to the action at training time. Soft Actor-Critic (SAC) [27] explores with entropy regularization. Other methods of exploration include adversarial self-play [28] and parameter noise [29].

2) Open Problems: The search for an efficient method for exploration in a continuous high-dimensional action space such as robotics still remains challenging. Although ϵ-greedy

[24] is one of the most commonly used methods for exploration, it has several weaknesses. One problem is that it treats all actions (when acting randomly) equivalently. Therefore, the ϵ-greedy strategy is unguided, too naive, and does not explore areas with promising actions. For on-policy algorithms, the amount of randomness depends greatly on initial conditions and training procedure. During training, the scale of randomness is reduced due to the update policy rule favoring more exploitation. As a result, the policy might be trapped in local optima. For deterministic policies, noise is added to their actions during training time, and the scale of noise might be reduced to get more high-quality training time. This approach will become insufficient when facing sparse and deceptive reward problems. We also lack a useful benchmark that can be used to evaluate the performance of different approaches for exploration. Also, the performance of exploration strategies vary among environments and configurations, making it hard to quantify the true improvement. Safety when exploring with real robots is another concern. For instance, exploration strategy such as exploration in the face of uncertainty is very unsafe for fragile robots.

C. Generalization and Reproducibility

Generalization is a critical stepping stone that many researchers hope that RL algorithms can achieve. For future robots facing a variety of complex real-world environments, the ability to function in a diverse environments is expected. Unfortunately, most of RL algorithms are trained with tuned hyperparameters specifically for a certain task or a small set of tasks, and they often fail with a novel task or environment. On the other hand, reproducibility is an underestimated issue in RL and not many researchers have tried to dig into the issue. It is not easy to reproduce results from many state-of-the-art papers since the implementation details were either missing or incomplete. The situation is even worse when coupled with the instability that RL algorithms currently suffer.

1) Brief Review: There are two main directions currently used to investigating the generalization of RL algorithms. The first approach is similar to robust control in control theory when policies are designed so that they can still function with environment variations by scarifying performance in other environments. In this direction, [30] learned a policy to maximize the conditional value at risk over a distribution of environments and [31] maximized expected reward over the subset of environments with lowest expected reward. [32] used adversarial training for learning a robust policy. The second approach is similar to adaptive control, trying to adapt to the the current environment such as in [33]. A number of algorithms [34] [35] used trajectories sampled from the ongoing environment as a mechanism to identify the environment, triggering the autonomous adaption of the policy. About reproducibility in RL, one of the best papers digging into this issue is [36] with an analysis of the reliance of the performance on a number of factors.. Network structure is one of factors that can significantly impact the performance in RL algorithms such as in case of TRPO and DDPG [37]. Random seed is another factor having a large effect on performance. Reported performance cannot be reliable if it is only tested on few numbers of random seeds. As shown in Figure 4, when TRPO runs on the same set of hyperparameters and with two different random seeds, the performances in two cases were significantly different. [36] also compared performance in a number of other factors such as environments, implementation (codebases), reward scales. For all tested factors, the performance varied significantly. To improve robustness, some research tried to close the control loop with visual feedback [38], or optimize hyper-parameteres using genetic algorithms [39].

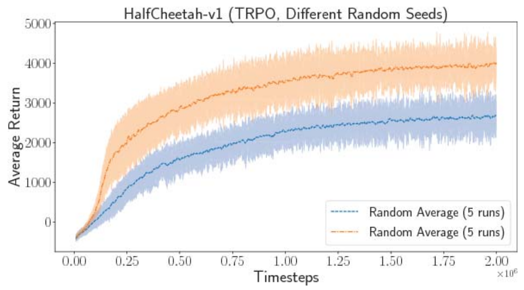

Fig. 4: TRPO's performance varied with random seeds (credit [36]).

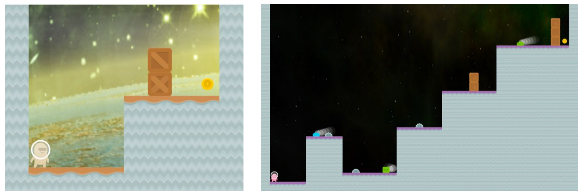

Fig. 5: CoinRun Environment (credit [40])

2) Open Problems: We currently do not have an effective benchmark to evaluate the generalization of RL algorithms. RL algorithms need something like the ImageNet dataset in supervised learning for testing generalization in a diverse range of tasks. For such a testbed for measuring generalization, we also need to clearly define set of tasks, comparison metrics, and baselines so that we can quantify fairly the generalization of RL algorithms. In an effort to quantify generalization in RL algorithms, OpenAI recently released CoinRun (Figure 5), which is an initial benchmark for measuring generalization. It was also shown in [40], techniques that are commonly used in supervised learning against over-fitting such as dropout, regularization and batch normalization, could improve generalization in RL. Combating reproducibility in machine learning in general is hard and it is even more challenging in RL for continuous environments such as in robot due to a higher instability. Beside the need for more robust RL algorithms to hyperparameters, we would possibly need agree on proper experimental methodologies and proper evaluation methods and metrics to address the problem. Effective tools for documenting changes during experiment setup also useful for improving reproducibility. It is also necessary to have a standard set of environments so that reproducibility can be verified fairly.

V. Future Directions

Possibly the strongest motivation for future developments in the field is how to efficiently take deep RL algorithms to the real world to solve practical applications. Therefore, we need to know what does it take in order to solve real-world problems. From our perspective, agents/robots must learn much faster and more efficiently. The future holds great potential for several branches for research including model-based learning, learning from prior trained tasks, and transfer learning and/or domain adaptation [41] [42].

Model-based learning has the biggest advantage of being sample-efficient and there existed interesting research in this direction to predict the future. In the context of Atari games, [43] used deep network architecture to successfully predict over 100 steps of future frames. As this approach is visually-based, it has the potential to generalize to other visually rich RL problems. Another research [44] used recurrent neural networks to make temporally and spatially coherent predictions for hundreds of time-steps into the future to improve exploration in Atari and some 3D games. In the context of robot manipulation, a recent paper [45] introduced Stochastic Adversarial Video Prediction (SAVP) - a Generative Adversarial Network (GAN) [46] and Variational Autoencoder (VAE) [47] variant, which can predict several hundred frames despite being trained to predict 10 future frames. Another interesting idea was introduced in [48] when model learning and planning were integrated to form one end-to-end training procedure. This approach addressed the previous problem when the estimated models were not consistent with the real model, resulting in poor performance in planning. However, these recent research on model-based approach in our perspective are just starting to work in rich environments and there is still a long way to go.

The ability to learn from other tasks is still very difficult for current RL algorithms. There is still a big gap between even state-of-the-art RL algorithms and humans in terms of sample efficiency when learning a novel skill. The reason why humans are much quicker learners might be that we do not learn from scratch. Instead, we can reuse our past knowledge to learn a new skill much more efficiently. Model-based learning approach can also help in this situation, thanks to its more potential transferability and generality. Model of environment can be reused for various tasks, which might be governed by the same laws of physics. [49] used a medium-sized neural network to approximate the dynamics and then used model predictive control (MPC) to produce a stable performance to accomplish various complex locomotion tasks in MuJoCo [50]. In this paper, they also combined model-based and model-free approach by using a learned model-based controller to generate roll-outs for fine-tuning using model-free learning. The combination resulted in accelerating learning with a gain of 3-5x more sample efficiency. A different approach rather than approximating the dynamics is to use multi-task learning to reuse skills [51]. The interesting in this work was learning on various tasks actually had better performance compared to learning in a single-task setting. By having the same big neural network for several tasks instead of using a smaller network for each task, the performance was significantly better for several tasks.

Transfer learning tries to use experience from one set of tasks for faster learning and better performance on a new task. Transfer learning from tasks trained on simulators is in particularly tempting as the relative cheap resources needed. Another recent approach is using domain adaptation to perform transfer learning [42] between related Atari games. The idea was to first train a policy in a source game in an actor-critic fashion and transfer state representation in this domain to initialize the policy network for the target domain. This approach resulted in a significant boost in sample efficiency. [52] performed parallel learning between simulated and real robots by introducing additional alignment rewards that encourage both agents in two domains to have similar distributions over visited states.

Inverse RL [53] is also a promising future direction, which can solve the nightmare of designing reasonable reward functions. Features autonomously learned by convolutional neural networks have revolutionized the world of computer vision, we can also expect the same with the ability to learn reward functions from expert policies.

VI. Conclusion

In this paper, we described the current picture of RL algorithms used in robot manipulation. Despite significant advancements of RL in simulated domains such as games, its potentially great influence on real robot applications is still limited. Best RL algorithms currently can acquire high proficiency in domains with simple and known rules such as board games (Go, Chess). When facing new territories with unknown dynamics, robots can only perform simple manipulation tasks given enough samples to learn. Compared to the variety of tasks that humans can learn and perform efficiently in a much shorter time, there is still a long way to go until we can build truly intelligent robots. The community is currently divided into multiple research directions, but we believe the solution can be reached by combining the advantages of some or all of them, or even maybe novel types of algorithms. However, from our point, RL in robot manipulation will have a promising future ahead. We believe that in order to build truly intelligent robots in the future, we would need something like RL algorithms.

References

[1] H. M. La, R. Lim, and W. Sheng, "Multirobot cooperative learning for predator avoidance," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 52–63, 2015.

[2] M. Rahimi, S. Gibb, Y. Shen, and H. M. La, "A comparison of various approaches to reinforcement learning algorithms for multi-robot box pushing," in *Intern. Conf. on Engineering Research and Applications*. Springer, 2018, pp. 16–30.

[3] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," 2018.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The Intern. J. of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[10] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Intern. Conf. on Machine Learning*, 2016, pp. 1928–1937.

[11] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Intern. Conf. on Machine Learning*, 2015, pp. 1889–1897.

[12] 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.

[13] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Intern. Conf. on Machine Learning*, 2016, pp. 2829–2838.

[14] S. Levine and V. Koltun, "Guided policy search," in *Intern. Conf. on Machine Learning*, 2013, pp. 1–9.

[15] M. Deisenroth and C. Edward Rasmussen, "Pilco: A model-based and data-efficient approach to policy search." 01 2011, pp. 465–472.

[16] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Robotics: Science and Systems*, 2011.

[17] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Robotics: Science and Systems*, 2018.

[18] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.

[19] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE Intern. Conf. on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[20] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," *arXiv preprint arXiv:1706.04652*, 2017.

[21] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell, "Towards adapting deep visuomotor representations from simulated to real environments," *CoRR, abs/1511.07111*, 2015.

[22] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The Intern. J. of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[23] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[24] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," in *Annual Conf. on Artificial Intelligence*. Springer, 2010, pp. 203–210.

[25] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Intern. Conf. on Machine Learning*, 2015, pp. 1889–1897.

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

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[28] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," *arXiv preprint arXiv:1703.05407*, 2017.

[29] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," *arXiv preprint arXiv:1706.01905*, 2017.

[30] A. Tamar, Y. Glassner, and S. Mannor, "Optimizing the cvar via sampling." in *AAAI*, 2015, pp. 2993–2999.

[31] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards generalization and simplicity in continuous control," in *Advances in Neural Information Processing Systems*, 2017, pp. 6550–6561.

[32] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *arXiv preprint arXiv:1703.02702*, 2017.

[33] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.

[34] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "Meta-learning with temporal convolutions," *arXiv preprint arXiv:1707.03141*, 2017.

[35] F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang, "Learning to learn: Meta-critic networks for sample efficient learning," *arXiv preprint arXiv:1706.09529*, 2017.

[36] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *arXiv preprint arXiv:1709.06560*, 2017.

[37] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup, "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control," *arXiv preprint arXiv:1708.04133*, 2017.

[38] H. Nguyen, H. M. La, and M. Deans, "Deep learning with experience ranking convolutional neural network for robot manipulator," *arXiv preprint arXiv:1809.05819*, 2018.

[39] A. Sehgal, H. M. La, S. J. Louis, and H. Nguyen, "Deep reinforcement learning using genetic algorithm for parameter optimization," *Submitted for Intern. Conf. on Robotic Computing (IRC)*, 2019.

[40] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," *arXiv preprint arXiv:1812.02341*, 2018.

[41] H.-J. Ye, X.-R. Sheng, D.-C. Zhan, and P. He, "Distance metric facilitated transportation between heterogeneous domains." in *IJCAI*, 2018, pp. 3012–3018.

[42] T. Carr, M. Chli, and G. Vogiatzis, "Domain adaptation for reinforcement learning on the atari," *arXiv preprint arXiv:1812.07452*, 2018.

[43] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, 2015, pp. 2863–2871.

[44] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed, "Recurrent environment simulators," *arXiv preprint arXiv:1704.02254*, 2017.

[45] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," *arXiv preprint arXiv:1804.01523*, 2018.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[47] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[48] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto *et al.*, "The predictron: End-to-end learning and planning," *arXiv preprint arXiv:1612.08810*, 2016.

[49] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE Intern. Conf. on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

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

[51] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE Intern. Conf. on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[52] M. Wulfmeier, I. Posner, and P. Abbeel, "Mutual alignment transfer learning," *arXiv preprint arXiv:1707.07907*, 2017.

[53] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *ICML*, 2000, pp. 663–670.