iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VIII **Month of publication:** August 2026

DOI: https://doi.org/10.22214/ijraset.2026.84647

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VIII Aug 2026- Available at www.ijraset.com

A Survey of Deep Reinforcement Learning Approaches in Robotic Manipulation

Sachidananda M H[1], Prashanth Kumar R[2], Darshan P R[3]
Department of Computer Science, PES Institute of Advanced Management Studies

Abstract: ***Neural-network-based reinforcement learning has driven a broad set of breakthroughs across many application areas. In the specific case of robotic manipulation, these methods raise the prospect of machines acquiring dexterous, human-like manipulation skills straight from visual input. This survey examines where reinforcement learning algorithms currently stand within this field. It lays out the core theory and terminology involved, and highlights the principal obstacles that still restrict these algorithms from being deployed on real-world robotic problems. The paper closes with the authors' outlook on promising directions for continued research.***

I. INTRODUCTION AND APPROACH TO THE REVIEW

Interest in reinforcement learning (RL) has surged over the past several years, fueled by notable achievements such as outperforming top human players in Atari games and in the game of Go. Applied to robotic manipulation, RL provides both a conceptual framework and practical tools for teaching robots dexterous, end-to-end manipulation directly from raw pixel data. While early results in this space were encouraging, they also exposed a set of persistent difficulties that arise when trying to apply RL to real robotic tasks. This survey sets out to capture the authors' view of where RL research stands with respect to robotic manipulation — covering foundational concepts, notable findings, unresolved challenges, and the authors' perspective on where the field should head next. The review's timeframe begins in 2013, since earlier RL work relevant to robotics had already been extensively documented in an existing survey. Given the lack of any comprehensive review covering RL for robotic manipulation from 2013 onward — despite a number of noteworthy results in that period —the authors set out to fill that gap, focusing specifically on manipulator applications. Their literature search drew on IEEE Xplore, Google Scholar, and arXiv, applying the following inclusion criteria: (a)publication in English; (b) publication year 2013 or later; (c) for work that appeared in both conference and journal form, only the journal version was included when it differed meaningfully from the conference paper; and (d) direct relevance to deep RL as applied to robot manipulation. Searches combined the terms "reinforcement learning,""deep reinforcement learning," and "robot manipulation," ultimately was narrowing the pool down to 42 relevant papers.

The remainder of the paper proceeds as follows: Section II introduces core RL concepts and vocabulary; Section III presents a taxonomy of RL algorithm families; Section IV narrows the focus to robot manipulation specifically; Section V lays out the authors' view on future research directions; and Section VI wraps up the survey.

II. FOUNDATIONAL CONCEPTS AND VOCABULARY

In the typical RL setup, an agent continually interacts with its environment (see Fig. 1). Aside from the agent and environment themselves, an RL system is generally built from four components: a policy, a reward signal, a value function, and — optionally — a model of the environment.

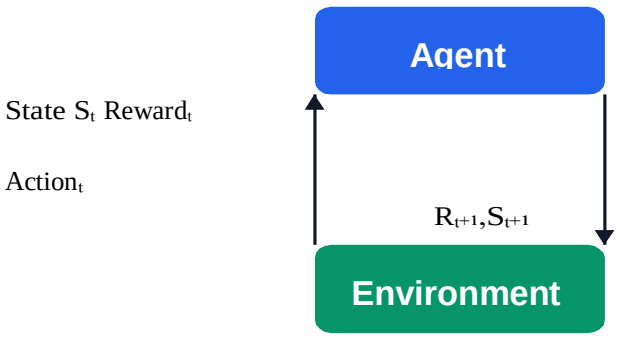


Fig.1(recreated):The agent–environment interaction loop in reinforcement learning

A state captures everything there is to know about the world data given moment nothing relevant is hidden from it. A policy dictates how the agent behaves at any point in time, mapping observed states onto actions. This mapping may be probabilistic, yielding a distribution over actions given the state ($a = \pi(.|s)$), or it may be deterministic ($a = \mu(s)$).

The reward signal encodes what the RL problem is actually trying to achieve. At every time step the environment returns a single number — the reward — to the agent, computed as a function of the current state and the chosen action, $r = R(s, a)$. The agent's overarching aim is to maximize the total reward accumulated over time. Formally, this means maximizing the expected discounted return G_t, where the discount factor γ lies in [0, 1]:

$G_t=R_{t+1}+\gamma R_{t+2}+\gamma ZR_{t+3}+...$

A model of the environment lets the agent anticipate how that environment will respond to its actions. Environment dynamics are captured by a distribution p, which may take a deterministic form ($s' = p(s,a)$) or a stochastic one ($s' = p(.|s,a)$).

The Markov Decision Process (MDP) is the classical mathematical framework for sequential decision-making, capturing how actions shape not only the reward received immediately but also the states and rewards that follow. An MDP therefore has to reconcile immediate reward against reward deferred to later.

It is defined by:

- A set of states, S
- A set of actions, A
- A reward function, $R(s,a)\in R$
- An environment transition distribution, $p(s',r|s, a)$

A value function captures how favorable a given state (or state-action pair) is over the long run, under some policy π. This yields two related functions: the state-value function $V^\pi(s)$, and the action-value function $Q^\pi(s,a)$:

$V^\pi(s)=E_{a\sim\pi}[R(\tau)|s_t=s]Q^\pi(s,a)=E_{a\sim\pi}[R(\tau)|s_t=s,a_t=a]$ The optimal value function corresponds to the policy that performs best:

$V(s)= \max_\pi E_{a\sim\pi}[R(\tau)|s_0=s]Q(s,a)= \max_\pi E_{a\sim\pi}[R(\tau)|s_0=s, a_0= a]$

$OnceQ(s,a)isknown,theoptimalactionatanystatesfollowsdirectly,givingtheoptimalpolicy\pi: a(s) = \arg\max_a Q(s,a)$

All fou rvalue functions above satisfy the Bellman equation:

$V^\pi(s)=E_{s'\sim\pi}[r(s,a)+\gamma V^\pi(s')]Q^\pi(s,a)=E_{s'\sim\pi}[r+E_{a'\sim P}[Q(s',a')]]V(s)=E[r(s,a)+\gamma V(s')]$ $Q(s,a) = E_{s'\sim\pi}[r + \gamma \max_a Q(s',a')]$

III. A TAXONOMY OF RL ALGORITHMS

Producing as single taxonomy that neatly captures every RL algorithm applied to manipulation is genuinely difficult. The authors restrict their discussion to two major dividing lines — model-based versus model-free, and policy-based versus value-based — along with the trade-offs each entails. (OpenAI's "Spinning Up" resource offers considerably fuller algorithm taxonomy, referenced in Fig. 2.)

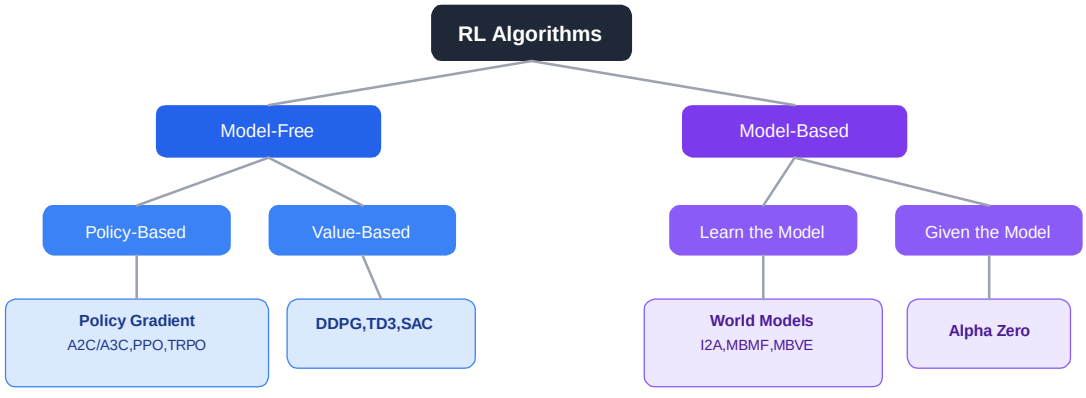


Fig.2 (recreated): A simplified taxonomy of RLalgorithm families referenced in the survey (BasedonthebranchstructurepopularizedbyOpenAI's"SpinningUp"resource)

A. *Model-Based versus Model-Free*

Algorithms can be separated according to whether the agent has, or learns, a model of its environment. Possessing such a model lets the agent look ahead — anticipating state transitions and future rewards before acting, and thereby foreseeing the consequences of each candidate action. Where the model is accurate, this yields a substantial sample-efficiency advantage over model-free approaches. In practice, however, a reliable model is often unavailable, and learning one accurately is difficult. A key risk is that a learned model can introduce systematic bias: an agent might excel within its learned model yet perform poorly, or sub-optimally, when acting in the actual environment. Model-free methods instead learn value functions purely from direct interaction with the environment, depending heavily on the reward signal for thislearning.Becausetheytendtobesimplertoimplementandtune,model-freemethodscurrentlyseefar more use than their model-based counterparts.

B. *Policy-Based versus Value-Based*

Value-based algorithms aim to approximate the optimal action-value function $Q(s,a|\theta)$ directly, typically doing so off-policy — meaning the behavior policy used to gather training data can differ from the policy that is actually being evaluated and improved (the "estimation policy"). The best action is then recovered via a = arg max_a Q(s,a). Policy-based algorithms instead parameterize the policy itself, $\pi(s,a|\theta)$, and optimize θ directly — either via gradient ascent on an objective $J(\pi)$, or by maximizing local surrogates for J. These methods are usually on-policy: they evaluate a policy's value using data generated by that same policy for control, which makes them less sample-efficient, since only data from the current policy version can be used. Optimizing the policy directly (as policy-based methods do) tends to produce more stable, reliable learning. Q-learning-style approaches, by contrast, only estimate Q indirectly via an objective function, which introduces several possible failure modes and makes them comparatively less stable. Their major redeeming advantage, though, is far greater sample efficiency, since they can reuse previously collected data much more effectively — a property that matters enormously for real robot deployments.

IV. APPLYING RL TO ROBOT MANIPULATION

Robotics problems generally present RL with continuous, high-dimensional state and action spaces. Manipulation tasks in particular make data collection costly and slow, and the resulting experience is often noisy and hard to replicate consistently. Because robot states are frequently unobservable or only partly observable, robotics RL is commonly framed as a partially observable MDP. Model-based methods, especially, must therefore be resilient to substantial model uncertainty. The authors identify three issues, from their own vantage point, that most limit RL's applicability to real robotic problems: sample inefficiency; the exploration-exploitation trade-off; and generalization together with reproducibility.

A. *Sample Inefficiency*

Sample inefficiency stands out as perhaps the single biggest obstacle to applying RL in manipulation settings — so much so that even leading algorithms can remain impractical because of it. Several factors drive this problem: many algorithms attempt to learn tasks entirely from scratch, requiring large volumes of data ;existing methods are still not very effective at extracting maximal value from the data they already have; some on-policy algorithms demand fresh data at every single update; and, more generally, robotic data collection itself is slow.

1) Summary of Prior Work: Evolutionary algorithms rank among the least sample-efficient approaches, since they forgo gradient information, though their final performance can be competitive. One evolution-strategies method matched benchmark Atari performance using 3–10x more data than a comparison method. The asynchronous actor-critic method A3C achieved greater data efficiency, surpassing that same benchmark using only a multi-core CPU rather than a GPU. Policy gradient methods rank next in sample efficiency, followed by replay-buffer-based Q-value estimation methods such as DDPG and Normalized Advantage Functions (NAF). Model-based algorithms lead the pack in data efficiency, since theyconstruct an environment model and train the policy against that model rather than against raw interaction data. Guided Policy Search achieves strong data efficiency through trajectory optimization that steers policy learning away from poor local optima. Currently, the "shallow" model-based method PILCO holds the lead; one study using PILCO learned a block-stacking task in roughly four minutes, a figure further reduced to 90 seconds via knowledge transfer.

2) Remaining Challenges: Improving data efficiency requires both gathering more data and using existing data more effectively. Running multiple robots in parallel is one route to more data (Fig. 3). Synthetic data — often sourced from simulation — offers another route, and several studies have pursued it; here, closing the gap between simulated and real-robot data is essential if the synthetic data is to be genuinely useful. That gap has been quantified for a grasping task specifically so that it can be actively

minimized during training. One study mapped synthetic images to realistic ones using a deep learning architecture; another applied progressive networks to reuse low-level visual features when transferring to high-level tasks. There is also a broader need for shared-data infrastructure, similar to what supervised learning enjoys through large public datasets — though robotics data tends to be tied closely to particular robot platforms and configurations, so a mechanism for making such data portable across platforms would be valuable. Ultimately, more efficient algorithms will also be needed; model-based approaches may be among the most promising routes toward unlocking sample efficiency.

(The original paper includes a photo of a multi-robot data-collection setup at this point, credited to Levine et al. As that photo is copyrighted material from the cited source, it is not reproduced here — see the reference list, item [22], for the original image.)

B. Exploration and Exploitation

Because an RL agent must repeatedly decide how to act given its current state, the exploration-versus-exploitation dilemma arises at every step. Exploration yields additional knowledge that can improve future decision-making, while exploitation selects the currently best-known action to maximize near-term reward. Effective strategies typically trade some short-term reward for greater long-term gain, requiring a careful balance between the two.

1) Summary of Prior Work: Deep Q-Networks (DQN) rely on ε-greedy exploration, taking a random action with probability ε and the value-maximizing action otherwise. Variants include decaying ε-greedy (where ε shrinks over training) and adaptive schemes that adjust ε based on temporal-difference signals. Vanilla policy gradient, Trust Region Policy Optimization (TRPO), and Proximal Policy Optimization (PPO) all explore by sampling from the current stochastic policy. DDPG instead trains a deterministic policy off-policy, adding noise to actions at training time. Soft Actor-Critic(SAC) achieves exploration through entropy regularization, while other approaches include adversarial self-play and parameter-space noise.

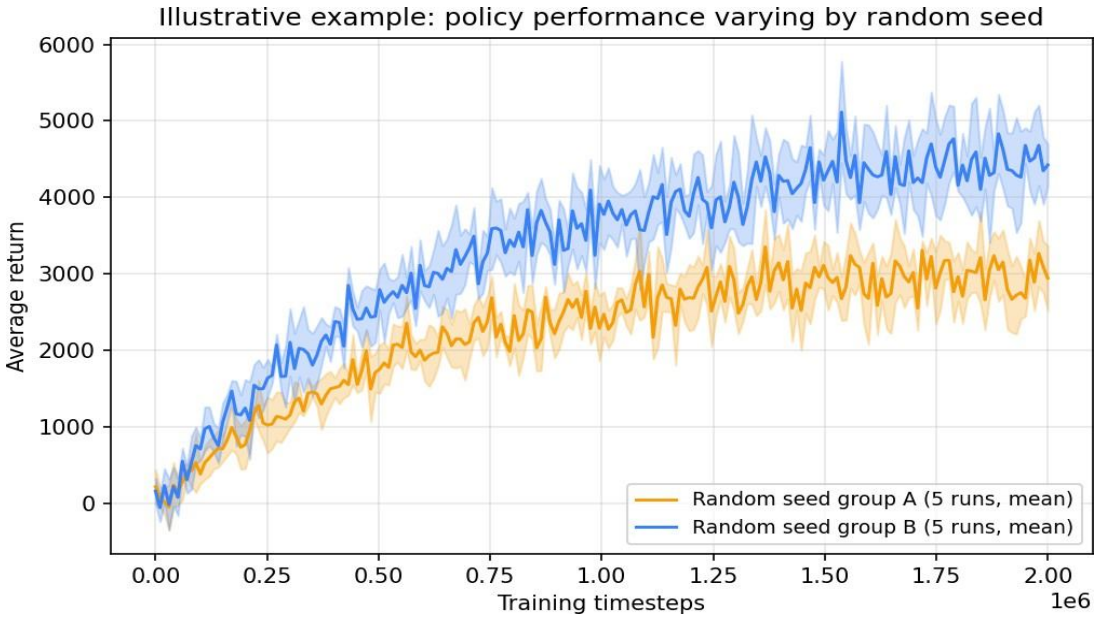
2) Remaining Challenges: Efficient exploration in continuous, high-dimensional action spaces — as found in robotics — remains an open problem. Although ε-greedy is among the most widely used exploration schemes, it has notable weaknesses: it treats all randomly chosen actions as equally valuable, making it naive and poorly guided toward promising regions of the action space. For on-policy methods, the degree of randomness depends heavily on initial conditions and the training regime; as training proceeds, policy updates progressively favor exploitation, which risks trapping the policy in local optima. Deterministic policies instead add noise directly to actions during training, typically reducing that noise over time to obtain higher-quality behavior later on — an approach that breaks down for sparse or deceptive reward signals. The field also lacks a solid benchmark for comparing exploration strategies, and performance of any given strategy tends to vary substantially across environments and settings, making genuine improvements hard to measure. Safety is a further concern: some exploration strategies, such as those seeking out uncertain regions, are risky to use on physically fragile robots.

C. Generalization and Reproducibility

Generalization is widely seen as essential if RL is to have real impact, since future robots will need to operate across a wide variety of complex, real-world settings. Yet most current RL algorithms are trained and tuned for one specific task, or a narrow set of tasks, and tend to fail when faced with anything novel. Reproducibility is a separate, and comparatively under-examined, issue: results from many state-of-the-art papers are difficult to replicate because implementation details are frequently missing or incomplete, a problem compounded by RL's inherent training instability.

1) Summary of Prior Work: Two broad strategies exist for improving generalization. The first parallels robust control theory: policies are explicitly designed to tolerate environmental variation, accepting reduced performance in any single environment in exchange for broader robustness — examples include policies optimized against conditional value-at-risk across a distribution of environments, or against the worst-case subset of environments, or trained adversarially for robustness. The second strategy resembles adaptive control, aiming instead to adjust to whichever environment the agent currently faces; several algorithms achieve this by using trajectories sampled from the live environment to identify that environment and trigger corresponding policy adaptation.

On reproducibility specifically, one particularly influential analysis examined how RL performance depends on a range of implementation and experimental factors. Network architecture, for instance, was shown to substantially affect outcomes for both TRPO and DDPG. Random seed choice is another major factor: as illustrated in Fig. 4, TRPO trained with identical hyper parameters but different random seeds produced markedly different results, meaning performance figures based on only a handful of seeds cannot be trusted. That same analysis also compared performance across environments codebases, and reward scaling, finding meaningful variation along every dimension tested.



(The chart above illustrates the general phenomenon described in the original paper — that identical hyper parameters with different random seeds can yield very different training curves for TRPO. It uses representative synthetic data rather than reproducing the original figure, which is credited to Henderson et al. [36].) Proposed remedies include closing the control loop with visual feedback, and tuning hyper parameters via genetic algorithms.

2) Remaining Challenges: No effective benchmark yet exists for measuring RL generalization — something analogous to ImageNet's role in supervised learning is still needed, along with clearly defined task sets, comparison metrics, and baselines that would allow fair, quantitative comparison. OpenAI's CoinRun environment represents an initial step toward such a benchmark (its procedurally-generated platformer levels are not reproduced here, as CoinRun is OpenAI's own copyrighted software — see reference [40] for images of the environment). Work using CoinRun has also shown that supervised-learning regularization techniques — dropout, weight regularization, batch normalization — can improve RL generalization as well. Reproducibility is difficult in machine learning broadly, and even more so in continuous-control robotics RL given its higher inherent instability. Beyond building algorithms that are more robust to hyperparameter choices, the field will likely need consensus on standard experimental methodology, evaluation metrics, and better tooling for documenting experimental setup changes — along with a standardized set of environments against which reproducibility claims can be verified.

V. FUTURE DIRECTIONS

Arguably the strongest driving motivation for future work in this are a is figuring out how to bring deepRL out of simulation and into practical, real-world use — which in turn requires understanding what solving real-world problems actually demands. In the authors' view, robots and agents need to become dramatically faster and more efficient learners. Promising research avenues include model-based learning, reusing knowledge from previously learned tasks, and transfer learning or domain adaptation.

Model-based learning's chief appeal is its sample efficiency, and there has been notable work aimed at predicting future outcomes. In Atari environments, one deep network architecture successfully predicted more than 100 future frames; because this method is vision-based, it could plausibly generalize to other visually rich RL problems. Separately, recurrent networks have been used to generate temporally and spatially consistent predictions hundreds of steps into the future, improving exploration in Atari and certain 3D game environments. For robot manipulation specifically, one recent method — Stochastic Adversarial Video Prediction (SAVP), which combines a GAN with a VAE — could predict several hundred frames despite being trained on only ten-frame prediction targets. Another notable idea integrated model learning and planning into a single end-to-end training pipeline, addressing the common problem where a learned model diverges from reality and thereby degrades planning performance. Even so, in the authors' assessment, model-based methods are only beginning to work in genuinely rich environments, and considerable work remains.

Learning from prior tasks remains a major difficulty for today's RL algorithms; even the best current methods are still far less sample-efficient than humans at picking up new skills. Part of the reason humans learn so quickly may be that we rarely learn entirely from scratch — instead reusing prior knowledge to accelerate new learning. Model-based approaches may help here too, given their greater potential for transfer and generality, since an environment model built around shared physical laws can be reused across tasks. One study used a moderately sized neural network to approximate system dynamics, then applied model predictive control (MPC) to achieve stable performance across a range of complex locomotion tasks in the MuJoCo simulator; that same work combined model-based and model-free learning by using a model-based controller to generate rollouts subsequently fine-tuned with model-free methods, yielding a 3–5x sample-efficiency improvement. A different strategy — rather than approximating dynamics directly — uses multi-task learning to share skills across tasks; interestingly, this work found that training on multiple tasks simultaneously, using one larger shared network, actually outperformed training separate smaller networks on individual tasks.

Transfer learning seeks to leverage experience from one task set to learn faster, and perform better, on a new task. Transferring from simulation-trained policies is especially attractive given the relatively low cost of simulated data. One recent approach applied domain adaptation to transfer learning between related Atari games — training an actor-critic policy on a source game, then using its learned state representation to initialize a policy network for a target game — achieving a substantial sample-efficiency gain. Another approach ran parallel learning across simulated and real robots, introducing alignment rewards that encouraged both agents to visit similar state distributions. Inverse RL is likewise a promising avenue, potentially resolving the persistent difficulty of hand-designing suitable reward functions; just as convolutional networks transformed computer vision through automatically learned features, similar advances may be possible for learning reward functions directly from expert demonstrations.

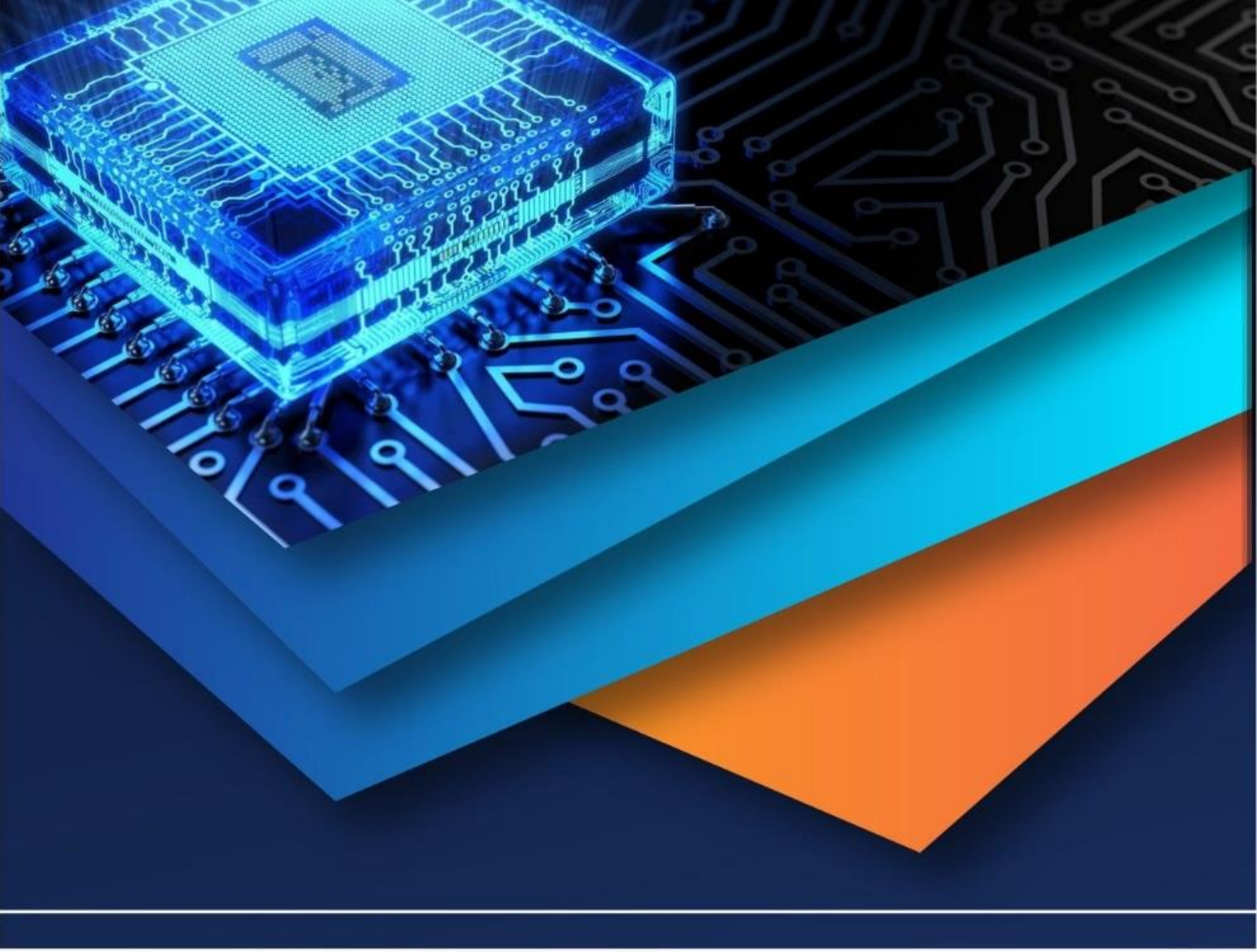
VI. CONCLUSION

This survey has outlined the current state of RL algorithms as applied to robot manipulation. Despite considerable progress in simulated domains such as games, RL's real-world impact on physical robots remains fairly limited. Today's best RL algorithms achieve strong performance in domains governed by simple, fully known rules, such as board games like Go and chess. When confronted with unfamiliar dynamics, robots can currently manage only simple manipulation tasks, and only given ample training samples. Relative to the breadth of tasks humans can learn quickly and perform well, there is still substantial ground to cover before truly intelligent robots become a reality. The research community remains split across several distinct directions, but the authors believe that progress will most likely come from combining the strengths of some or all of these approaches — or possibly from entirely new classes of algorithms not yet developed. Even so, the authors remain optimistic about RL's role in the future of robot manipulation, viewing algorithms in this family as a likely necessary ingredient for building genuinely intelligent robots.

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. Wier stra, and M. Riedmiller, “Playing atari with deep reinforcement learn ing,” 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. Roth¨orl, 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. disserta tion, 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, "Prox imal 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, "Reproducibil ity 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 facil itated 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. B¨ol¨oni, 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.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY