2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning

Jianlan Luo*[1], Zheyuan Hu*[1], Charles Xu[1], You Liang Tan[1], Jacob Berg[2], Archit Sharma[3], Stefan Schaal[4], Chelsea Finn[3], Abhishek Gupta[2] and Sergey Levine[1]

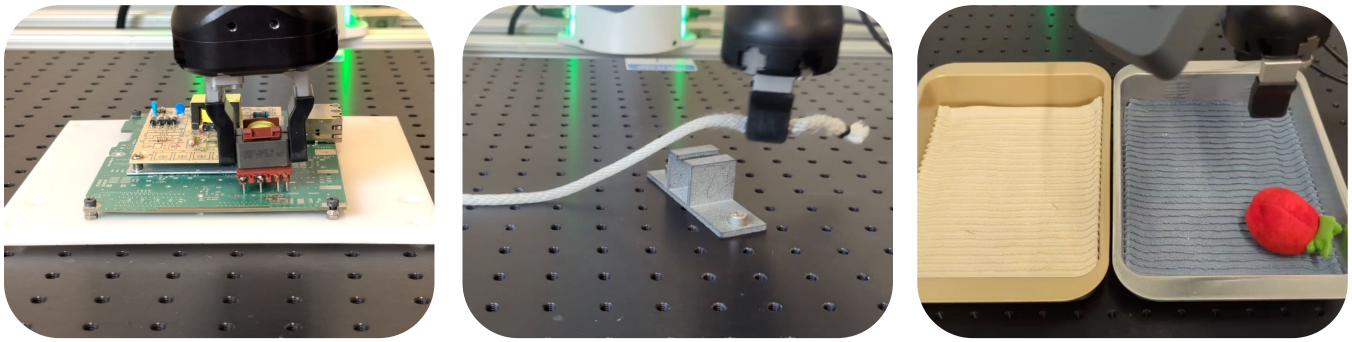

Fig. 1: Depiction of various tasks solved using SERL in the real world. These include PCB board insertion (left), cable routing (middle), and object relocation (right). SERL provides an out-of-the-box package for real-world reinforcement learning, with support for sample-efficient learning, learned rewards, and automation of resets.

***Abstract*— In recent years, significant progress has been made in the field of robotic reinforcement learning (RL), enabling methods that handle complex image observations, train in the real world, and incorporate auxiliary data, such as demonstrations and prior experience. However, despite these advances, robotic RL remains hard to use. It is acknowledged among practitioners that the particular implementation details of these algorithms are often just as important (if not more so) for performance as the choice of algorithm. We posit that a significant challenge to the widespread adoption of robotic RL, as well as the further development of robotic RL methods, is the comparative inaccessibility of such methods. To address this challenge, we developed a carefully implemented library containing a sample efficient off-policy deep RL method, together with methods for computing rewards and resetting the environment, a high-quality controller for a widely adopted robot, and a number of challenging example tasks. We provide this library as a resource for the community, describe its design choices, and present experimental results. Perhaps surprisingly, we find that our implementation can achieve very efficient learning, acquiring policies for PCB board assembly, cable routing, and object relocation between 25 to 50 minutes of training per policy on average, improving over state-of-the-art results reported for similar tasks in the literature. These policies achieve perfect or near-perfect success rates, extreme robustness even under perturbations, and exhibit emergent recovery and correction behaviors. We hope these promising results and our high-quality open-source implementation will provide a tool for the robotics community to facilitate further developments in robotic RL. Our code, documentation, and videos can be found at https://serl-robot.github.io/**

*Equal contribution
[1]Department of EECS, University of California, Berkeley
[2]Department of Computer Science, University of Washington
[3]Department of Computer Science, Stanford University
[4]Intrinsic Innovation LLC

I. Introduction

Considerable progress on robotic reinforcement learning (RL) over the recent years has produced impressive results, with robots playing table tennis [1], manipulating objects from raw images [2, 3, 4], grasping diverse objects [5, 6], and performing a wide range of other skills. However, despite the significant progress on the underlying algorithms, RL remains challenging to use for real-world robotic learning problems, and practical adoption has been more limited. We argue that part of the reason for this is that the implementation of RL algorithms, particularly for real-world robotic systems, presents a very large design space, and it is the challenge of navigating this design space, rather than limitations of algorithms *per se*, that limit adoption. It is often acknowledged by practitioners in the field that details in the implementation of an RL algorithm might be as important (if not more important) as the particular choice of algorithm. Furthermore, real-world learning presents additional challenges with reward specification, implementation of environment resets, sample efficiency, compliant and safe control, and other difficulties that put even more stress on this issue. Thus, adoption and further research progress on real-world robotic RL may well be bottlenecked on *implementation* rather than novel algorithmic innovations.

To address this challenge, our aim in this paper is to provide an open-source software framework, which we call **S**ample-**E**fficient **R**obotic reinforcement **L**earning (SERL), that aims to facilitate wider adoption of RL in real-world robotics. SERL consists of the following components: (1) a high-quality RL implementation that is geared towards real-world robotic learning and supports image observations

979-8-3503-8457-4/24/$31.00 ©2024 IEEE

and demonstrations; (2) implementations of several reward specification methods that are compatible with image observations, including classifiers and adversarial training; (3) support for learning "forward-backward" controllers that can automatically reset the task between trials [7]; (4) a software package that can in principle connect the aforementioned RL component to any robotic manipulator; and (5) an impedance controller design principle that is particularly effective for dealing with contact-rich manipulation tasks. Our aim in this paper is not to propose novel algorithms or methodology, but rather to offer a resource for the community to provide roboticists with a well-designed foundation both for future research on robotic RL, and other methods that might employ robotic RL as a subroutine. However, in the process of evaluating our framework, we also make a scientifically interesting empirical observation: when implemented properly in a carefully engineered software package, current sample-efficient robotic RL methods can attain very high success rates with relatively modest training times. The tasks in our evaluation are illustrated in Fig. 1: precise insertion tasks involving dynamic contact, deformable object manipulation with complex dynamics, and object relocation where the robot must learn without manually designed resets. For each of these tasks, SERL is able to learn effectively within 15 - 60 min of training per policy (in terms of total wall-clock time), achieving near-perfect success rates, despite learning policies that operate on image observations. This result is significant because RL, particularly with deep networks and image inputs, is often considered to be highly inefficient. Our results challenge this assumption, suggesting careful implementations of existing techniques, combined with well-designed controllers and carefully selected components for reward specification and resets, can provide an overall system that is efficient enough for real-world use.

II. Related Work

Our framework carefully combines existing RL methods into a complete, efficient, and ready-to-use robotic reinforcement learning system directly in the real world. Here, we summarize both related prior methods and systems.

Algorithms for real-world RL: Real-world robotic RL demands algorithms that are sample-efficient, can utilize onboard perception, and support easily specified rewards and resets. A number of algorithms have shown the ability to learn very efficiently directly in the real world [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], using variants of off-policy RL [20, 17, 21, 22], model-based RL [23, 24, 25, 26, 27], and on-policy RL [28]. These advances have been paired with improvements in rewards inference from visual observation through success classifiers [29, 30], foundation-model-based rewards [31, 32, 33], and rewards from videos [34, 35]. Additionally, algorithmic advancements in reset-free learning [2, 36, 37, 38, 39] have enabled autonomous training with minimal human interventions. While these advances are important, the contribution of this work is to provide a framework and software package to facilitate reinforcement learning in the real world with a ready-made choice of methods that can work well for a variety of tasks. We hope to lower the entry barrier for new researchers to build better algorithms and train robotic RL policies in the real world.

Software packages for RL: There are a number of packages [40, 41, 42, 43] for RL, though to our knowledge, none aim to directly address real-world robotic RL specifically. SERL builds on the recently proposed RLPD [44], which is an off-policy RL algorithm with SOTA sample efficiency. SERL is not a RL benchmark for different methods [45, 46, 47], although it could be adapted to be so. Rather, SERL offers a full stack pipeline for robot control, from low-level controllers, the interface for asynchronous and efficient training, to additional machinery for inferring rewards and training without resets. Together, SERL provides an off-the-shelf package to assist non-experts using RL and their physical robots to learn in the real world. Unlike prior libraries that aim to provide implementations of many methods – SERL offers a full "vertical" integration of components, whereas prior libraries focus on the "horizontal." SERL allows users to define their own tasks and success metrics directly in the real world, providing the software infrastructure for actually controlling and training robotic manipulators in these tasks.

Software for real-world RL: Several previous packages have proposed infrastructure for real-world RL: for dexterous manipulation [48], tabletop furniture assembly [49], legged locomotion [20], and peg insertion [50]. These packages are effective in narrow situations, either using privileged information, training setups such as explicit tracking [50, 48], pure proprioception [20], or limited to imitation learning. In SERL, we show a full stack system that can be used for a wide variety of robotic manipulation tasks without requiring complex privileged training setups as in prior work.

III. Preliminaries and Problem Statement

Robotic reinforcement learning tasks can be defined via an MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \rho, \mathcal{P}, r, \gamma\}$, where $\mathbf{s} \in \mathcal{S}$ is the state observation (e.g., an image in combination with the current end-effector position), $\mathbf{a} \in \mathcal{A}$ is the action (e.g., the desired end-effector pose), $\rho(\mathbf{s}_0)$ is a distribution over initial states, $\mathcal{P}$ is the unknown and potentially stochastic transition probabilities that depend on the system dynamics, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which encodes the task. An optimal policy π is one that maximizes the cumulative expected value of the reward, i.e., $E[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$, where the expectation is taken with respect to the initial state distribution, transition probabilities, and policy π.

While the specification of the RL task is concise and simple, turning real-world robotic learning problems into RL problems requires care. First, the sample efficiency of the algorithm for learning π is paramount: when the learning must take place in the real world, every minute of training comes at a cost. Sample efficiency can be improved by using effective off-policy RL algorithms [51, 52, 53], as well as incorporating prior data and demonstrations [54, 44, 55].

Additionally, there are many challenges in real-world robotic learning besides sample efficiency. For instance, the reward function r might depend on image observations,

which is difficult to specify manually. For episodic tasks where the robot resets to an initial state $\mathbf{s}_0 \sim \rho(\mathbf{s}_0)$ between trials, actually resetting the robot (and its environment) into one of these initial states requires further engineering effort.

Furthermore, the controller layer, which interfaces the MDP actions $\mathbf{a}$ (e.g., end-effector poses) to the actual low-level robot controls, also requires great care, particularly for contact-rich tasks where the robot physically interacts with objects in the environment. Not only does this controller need to be accurate, but it must also be safe enough that the RL algorithm can explore random actions during training.

SERL aims to provide ready-made solutions to each of these challenges, with a high-quality implementation of a sample-efficient off-policy RL method that can incorporate prior data, several choices for reward function specification, a forward-backward algorithm for learning resets, and a controller suitable for learning contact-rich tasks without damaging either the robot or objects in the environment.

IV. Sample Efficient Robotic Reinforcement Learning in the Real-World

Our software package, which we call **S**ample-**E**fficient **R**obotic reinforcement **L**earning (SERL), aims to make robotic RL in the real world accessible by providing ready-made solutions to the problems detailed in the previous section. This involves providing efficient vision-based reinforcement learning algorithms *and* the infrastructure needed to support autonomous learning. We note that the purpose of such an endeavor is not to propose novel algorithms or tools, but rather to develop a software package that anyone can use easily for robotic learning, without complex setup procedures and painful integration across libraries.

A. Core RL Algorithm: RLPD

There are several desiderata for the reinforcement learning algorithm to be deployed in a real-world setting: (1) it must be sample-efficient, (2) it must be able to incorporate prior data easily and then continue improving with further experience, (3) it must be simple to debug and build on for new users. To this end, we build on the recently proposed RLPD [44] algorithm. RLPD is an off-policy actor-critic reinforcement learning algorithm that develops upon Soft Actor-Critic [52] with some modifications to satisfy the desiderata above. RLPD makes three key changes: (i) high update-to-data ratio training (UTD), (ii) symmetric sampling between prior data and online data, such that half of each batch comes from prior data and half from the online replay buffer, and (iii) layer-norm regularization in critics. This method can train from scratch, or use prior data (e.g., demonstrations) to bootstrap learning. Each step of the algorithm updates the parameters of a parametric Q-function $Q_\phi(\mathbf{s}, \mathbf{a})$ and actor $\pi_\theta(\mathbf{a}|\mathbf{s})$ according to the gradient of their respective loss functions:

$$\mathcal{L}_Q(\phi) = E_{\mathbf{s},\mathbf{a},\mathbf{s}'}\Big[\big(Q_\phi(\mathbf{s},\mathbf{a}) - \big(r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{a}'\sim\pi_\theta}[Q_{\bar{\phi}}(\mathbf{s}',\mathbf{a}')]\big)\big)^2\Big]$$

$$\mathcal{L}_\pi(\theta) = -E_{\mathbf{s}}\left[E_{\mathbf{a}\sim\pi_\theta(\mathbf{a})}[Q_\phi(\mathbf{s},\mathbf{a})] + \alpha\mathcal{H}(\pi_\theta(\cdot|\mathbf{s})\right],$$

where $Q_{\bar{\phi}}$ is a *target network* [56], and the actor loss uses entropy regularization with an adaptively adjusted weight α [52]. For efficient learning, critics regularized with layer normalization perform multiple update steps per environment step, referred to as the update-to-date (UTD) ratio [44].

B. Reward Specification with Classifiers

Reward functions are difficult to specify by hand when learning with image observations. Our framework thus supports these three types of rewards: For tasks such as the PCB board assembly task in Fig. 1, hand-specified rewards based on the location of the end effector (under the assumption that the object is held rigidly in the gripper) are sufficient; for tasks require rewards to be deduced from images, the reward function can be provided by a binary classifier that takes in the image observation and outputs a binary reward corresponding to successful completion.

This classifier can be trained either with positive and negative examples collected by users, or via an adversarial method called VICE [29]. VICE circumvents the reward exploitation problem that arises with classifier-based rewards by assigning all states visited by the policy with negative labels and updating the classifier accordingly. The RL process is then analogous to a generative adversarial network (GAN) [57], with the policy acting as the generator and the reward classifier as the discriminator.

C. Reset-Free Training with Forward-Backward Controllers

When learning episodic tasks, the robot must reset the environment between task attempts. For example, when learning the object relocation task in Figure 1, each time the robot successfully moves the object to the target bin, it must then take it out and place it back into the initial bin. To remove the need for human effort in "resets", SERL supports "reset-free" training by using forward and backward controllers [58, 2]. In this setup, two policies are trained simultaneously using two independent RL agents, each with its own policy, Q-function, and reward function (specified via the methods in the previous section). The *forward* agent learns to perform the task, and the *backward* agent learns to return to the initial state(s). While more complex reset-free training procedures can also be possible [2], we find that this simple recipe is sufficient for learning object manipulation tasks like the repositioning skill in Figure 1.

D. Software Components

Environment adapters: SERL aims to be easily usable for many robot environments. Although we provide a set of Gym environment wrappers and robot environments for the Franka arm as starter guides, users can also use their own existing environments or develop new environments as they see fit. Thus, the library does not impose additional constraints on the robot environment as long as it is Gym-like [59] as shown in Fig. 3. We welcome contributions from the community to extend the support for readily deployable environment wrappers for other robots and tasks.

Asynchronous Updates: SERL includes options to decouple action inference and policy updates with a few lines of code

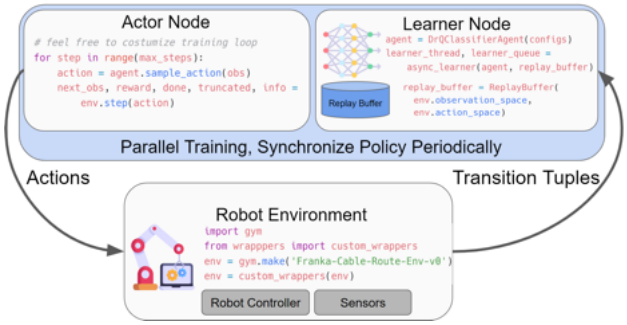

Fig. 3: Software architecture and real-world robot training example code. SERL runs three parallel processes, consisting of the actor, which chooses actions, and the learner node, which actually runs the training code, and the robot environment, which executes the actions from the actor and contributes data back to the learner.

as illustrated in Fig. 3. We found this beneficial in sample-efficient real-world learning problems with high UTD ratios. By separating actor and learner on two different processes, SERL not only preserves the control frequency at a fixed rate, which is crucial for tasks that require immediate feedback and reactions, such as deformable objects and contact-rich manipulations, but also reduces the total wall-clock time spend training in the real world.

E. Impedance Controller for Contact-Rich Tasks

Although our package should be compatible with any OEM robot controller as described in Sec. IV, we found the choice of controllers significantly impacts performance, especially for contact-rich manipulation such as the PCB insertion task in Fig. 1. An overly stiff controller could bend the fragile pins, whereas an overly compliant controller might struggle to move the object into position quickly.

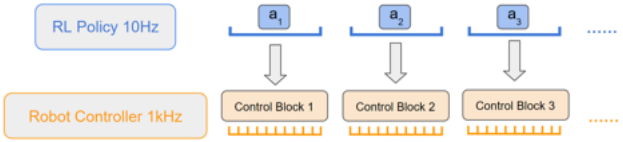

Fig. 4: A typical controller hierarchy for robotics RL. The output from the RL policy is tracked within a block of time by the downstream controller.

In robotic RL, the control system is often two-layered, where a high-level RL policy generates set-point actions at a lower frequency than the real-time controller it guides. For instance, an RL controller might issue commands at 10Hz, directing a lower-level impedance controller operating at 1kHz. This means a single RL decision influences 100 cycles of the lower-level controller. The goal of the impedance controller is to balance forces based on the equation $F = k_p \cdot e + k_d \cdot \dot{e} + F_{ff} + F_{cor}$, where $e = p - p_{ref}$, p is the measured pose, and p_{ref} is the target pose computed by the upstream controller, F_{ff} is the feed-forward force, F_{cor} is the Coriolis force. This setup functions similarly to a spring-damper system, where the distance to the target position (p_{ref}) can generate significant forces if not carefully managed, risking damage upon contact. Instead of compromising control precision by lowering gains, we bound e directly to a maximum value Δ, effectively capping the force to prevent hard collisions while maintaining accuracy. Then the force generated from the spring-damper system will be bounded to $k_p \cdot |\Delta| + 2k_d \cdot |\Delta| \cdot f$, where f is the control frequency.

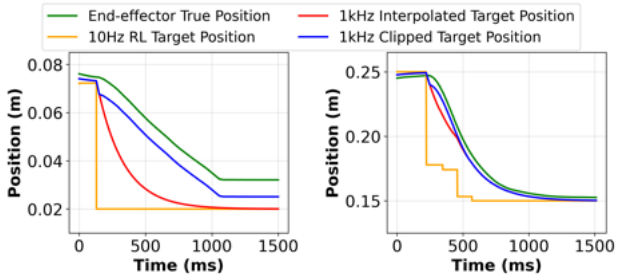

Fig. 5: Visualization of controller logs on the z-axis of the end-effector when commanded with different movements. The orange line is the commanded target (the output of RL), red is the smoothed target sent to the real-time controller, blue is the clipped target, and green is the robot position after executing this controller. **Left:** Command for the end-effector to engage with a hard surface, where the reference limiting clips the target to prevent a collision. **Right:** The command is a fast free-space movement, which our mechanism does *not* block, allowing fast motion to the target.

One might wonder if we could directly clip the RL policy's action output to small increments, on the order of micrometers for delicate tasks like PCB board manipulation. This might seem straightforward but can lead to prolonged or unstable learning processes compared to our approach due to the need for significantly more time steps to move the end-effector across the same distance. Instead, reference limiting at the real-time control layer addresses this issue without restricting the RL policy's range of action, allowing for unimpeded movement in free space as long as $M \cdot |\Delta| \geq |a|_{max}$, where M is the number of control time-steps inside a block, usually large (e.g., $M = 100$), as in Fig. 4. One might also ask whether it is possible to achieve the same result by using an external force/torque sensor. However, practical challenges such as sensor noise and calibration complexity make this approach less viable. Moreover, designing robot motions that both learn effectively and adhere to force constraints adds further complexity.

Our approach, which involves clipping the reference at the real-time control layer, is simple and effective, particularly for enabling RL in tasks requiring precise contact-rich manipulation. This technique was successfully implemented and verified with a Franka Panda robot, demonstrating that our controller can effectively limit forces during contact while allowing swift movements in free space as shown in Fig. 5. This strategy is not only applicable to Franka Panda robots but can also be adapted to any torque-controlled robot, showcasing its versatility in facilitating RL-based manipulation tasks.

F. Relative Observation and Action Frame

Selecting a suitable action space is crucial for expediting the training process and extending the policy's generalization ability to perturbations at test time. While SERL supports multiple action representations through a common RL interface, we found a convenient mechanism for representing observations and actions in the relative coordinate system.

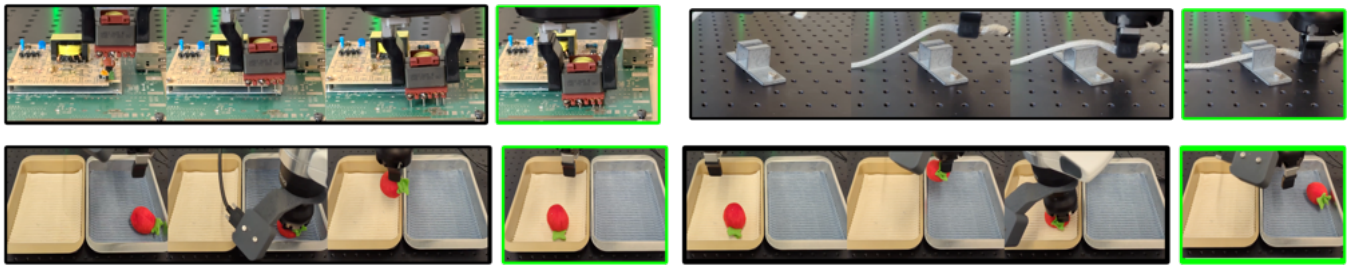

Fig. 6: Illustration of the robot performing each task: PCB Insertion (top left), Cable Routing (top right), Object Relocation - Forward (bottom left), and Object Relocation - Backward (bottom right). The green box indicates a state where the robot receives a high reward for completing the task.

To develop an agent capable of adapting to a dynamic target, we propose a training procedure that simulates a moving target without the need for physical movement. The target, for instance, the PCB insertion socket holes, is fixed relative to the robot base frame, and the reward can be specified using any of the standard methods provided in Sec. IV-B. At the beginning of each training episode, the pose of the robot's end-effector was randomized uniformly within a pre-defined area in the workspace. The robot's proprioceptive information is expressed w.r.t the frame of the end-effector's initial pose; the action output from the policy (6D twist) is relative to the current end-effector frame. This is equivalent to physically moving the target when viewed relatively from the frame attached to the end-effector. As a result, the policy can succeed even if the object moves or, as in some of our experiments, is perturbed in the middle of the episode.

V. Experiments

Our experimental evaluation aims to study how efficiently our system can learn a variety of robotic manipulation tasks, including contact-rich tasks, deformable object manipulation, and free-floating object manipulation. These experiments demonstrate the breadth of applicability and efficiency of SERL. We use a Franka Panda arm and two wrist cameras attached to the end-effector to get close-in views. Further details can be found at https://serl-robot.github.io/. We use an ImageNet pre-trained ResNet-10 [63] as a vision backbone for the policy network and connect it to a 2-layer MLP. Observations include camera images and robot proprioceptive information such as end-effector pose, twist, force, and torque. The policy outputs a 6D end-effector delta pose from the current pose, which is tracked by the low-level controller. The evaluation tasks are illustrated in Fig. 6 and below:

PCB insertion: Inserting connectors into a PCB board demands fine-grained, contact-rich manipulation with sub-millimeter precision. This task is ideal for real-world training, as simulating and transferring such contact-rich interactions can be challenging. At the beginning of each episode, the initial end effector pose is sampled uniformly from a starting region, as described in Table II.

Cable routing: This task involves routing a deformable cable into a clip's tight-fitting slot. This task requires the robot to perceive the cable and carefully manipulate it so that it fits into the clip while holding it at another location. This is particularly difficult for any method that relies on model-based control, or makes rigid-object assumptions, since both visual perception and handling of deformable objects is essential for access. Tasks of this sort often arise in manufacturing and maintenance scenarios. Similarly to the PCB task, the initial end effector pose is sampled uniformly within a starting region, as described in Table II.

Object relocation: This task requires moving a free-floating object between bins, requiring grasping and relocation. The intricacies of reward inference and reset-free training become especially pronounced in the manipulation of such free-floating objects. We define the forward task as picking up the object from the bin on the right side and placing it on the left, while the backward task moves the object back to the starting bin, undoing the forward task.

For each task, we initialize RL training from **20** teleoperated demonstrations using a Space Mouse. To confirm that demonstrations alone are insufficient to solve the task, we include a behavioral cloning (BC) baseline using **100** high-quality **expert** teleoperated demonstrations, roughly matching the total amount of data in the RL replay buffer when RL converges. Note that this is 5 times *more* demonstrations than the amount provided by our method. Both RL and BC demonstrations are collected using the initial end effector randomization scheme described in Table II. All training was done on a single Nvidia RTX 4090 GPU.

a) Results: We report the results in Table II, and show example executions in Fig. 6. We evaluated both BC and RL policies under the same conditions and protocols as detailed in Section V. Our RL policies achieve perfect success rates on all three tasks over all 100 trials. For the PCB insertion and cable routing task, our RL policies converge in under 30 minutes of real-world training, which includes all computation, resets, and intended stops. The free-floating object relocation task learns two policies (forward and backward), and total time amounts to less than an hour *per policy*. For the cable routing task and PCB insertion task, our policies outperform BC baselines by a large margin, despite training with 5x fewer demonstrations than BC, suggesting that demos alone are insufficient. We report the results in terms of success rate and cycle time in Fig. 7 and Fig. 8. The learned RL policies not only outperformed their BC counterparts by as much as 10x in terms of success rate but also improved on the cycle time of the initial human demonstrations by up to 3x.

b) Comparison to prior systems: While it's difficult to directly compare our results to those of prior systems due

Package	Task	Training time	Success rate	Demos	Shaping?	Vision?	Open-sourced?
Guided Policy Search[4]	Peg insertion	3 hours	70%	0	Yes	Yes	Yes
DDPGfD [60]	Peg/clip insertion	1.5-2.5 hours	97% / 77%	30	No	Yes	No
Visual Residual RL [19]	Connector insertion	Not mentioned	52% $\sim$ 100%	0	Yes	Yes	No
SHIELD [61]	Connector insertion	1.5 hours	99.8%	25	No	Yes	No
InsertionNet [62]	Connector insertion	40 mins	78.5% - 100%	0	Yes	Yes	No
SERL (Ours)	**PCB Insertion**	**20 mins**	**100%**	**20**	**No**	**Yes**	**Yes**

TABLE I: Comparison to results reported on similar tasks in prior work. The overall success rates for our method are generally higher, and the training times are generally lower, as compared to prior results. Note also that the PCB board assembly task, shown in Figure 1, has very tight tolerances, likely significantly tighter than the coarser peg and connector insertion tasks studied in the prior works.

Task	# of Demos	Image Input	Random Reset	Reward Specification	Bin Size	Training Time
PCB Component Insertion	20	2 wrist camera	True	Ground Truth	10cm $\times$ 10cm	20 mins
Cable Routing	20	2 wrist camera	True	Binary Classifier	20cm $\times$ 20cm	31 mins
Object Relocation (Forward-Backward)	20	1 wrist, 1 side camera	False	Binary Classifier	20cm $\times$ 30cm	105 mins

TABLE II: Task parameters: During demo collection for both BC and RL, as well as online training, each episode's initial end-effector pose resets uniformly at random within a fixed region for the PCB and Cable task, while the free-floating object relocation task resets above the center of each bin.

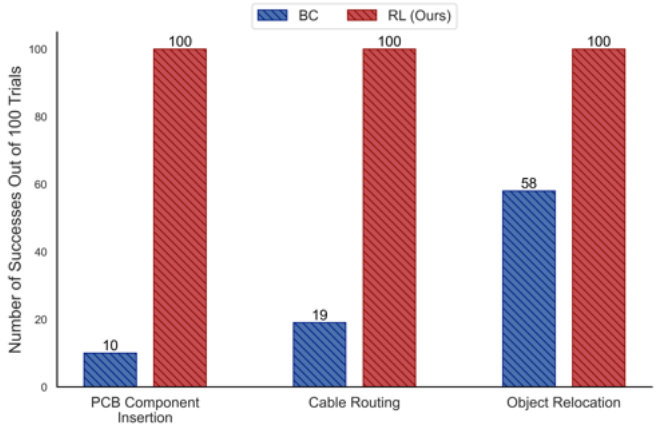

Fig. 7: Success rate comparisons: When evaluated for 100 trials per task, learned RL policies outperformed BC policies by a large margin, by **1.7x** for Object Relocation, by **5x** for Cable Routing, and by **10x** for PCB Insertion.

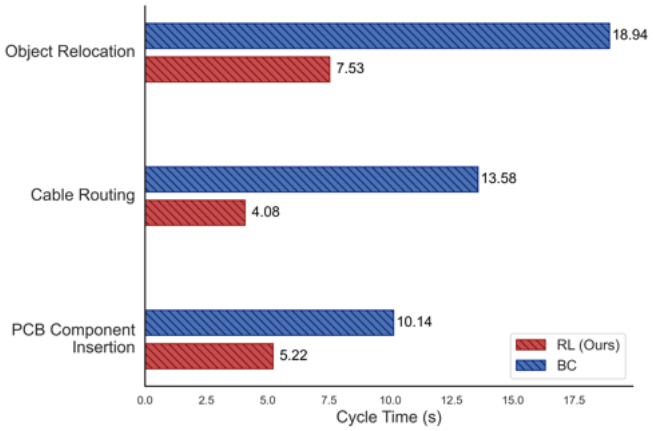

Fig. 8: Cycle time comparison: We recorded the average time taken for the robot to succeed in each task. RL policies are at least **2x** faster than BC policies trained with 100 high-quality human teleoperated demonstrations.

to numerous differences in the setup, lack of consistently open-sourced code, and other discrepancies, we provide a summary of training times and success rates reported for tasks that are most similar to our PCB board insertion task in Table II. We chose this task because similar insertion or assembly tasks have been studied in prior work, presenting challenges to precision, compliant control, and sample efficiency. Compared to prior works, our experiments do not use shaped rewards, which might require extensive engineering, though we do utilize a small amount of demonstration data (which some prior works eschew). The results reported in these prior works generally have lower success rates and/or longer training times, suggesting SERL matches or exceeds the performance of state-of-the-art methods in the literature on this task. The closest performance to ours in the work of Spector et al. [62] includes a number of design decisions and inductive biases specific to insertion, whereas SERL is generic and makes minimal task-specific assumptions. Although the components of SERL are all based on (recent) prior work, the state-of-the-art performance of this combination illustrates our main thesis: the details of how deep RL methods are implemented can make a big difference.

VI. Discussion

We introduced a software package aimed at making real-world robotic reinforcement learning more accessible for researchers and practitioners. SERL combines sample-efficient RL, automated reward design, environment resets, and a controller tailored for precise contact-rich manipulation tasks. Our experimental evaluation demonstrates SERL's efficiency in diverse manipulation tasks. However, there are several limitations: SERL doesn't cover all RL methods or non-manipulation tasks, and challenges such as reward specification and reset-free learning remain open problems. We hope that SERL provides a solid starting point for those exploring real-world robotic reinforcement learning.

Acknowledgments

This research was partially supported by Intrinsic Innovation LLC, the National Science Foundation under IIS-2150826, and ARO W911NF-21-1-0097. We would like to thank Rehaan Ahmad and Siri Gadipudi for participating in the discussion of this project.

References

[1] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters. "Learning to Play Table Tennis From Scratch Using Muscular Robots". In: *IEEE Trans. Robotics* 38.6 (2022), pp. 3850–3860. URL: https://doi.org/10.1109/TRO.2022.3176207.

[2] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. "Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention". In: *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 6664–6671. URL: https://doi.org/10.1109/ICRA48506.2021.9561384.

[3] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale". In: *CoRR* abs/2104.08212 (2021). arXiv: 2104.08212. URL: https://arxiv.org/abs/2104.08212.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[5] 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". In: *Int. J. Robotics Res.* 37.4-5 (2018), pp. 421–436. URL: https://doi.org/10.1177/0278364917710318.

[6] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*. Ed. by N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma. 2017. URL: http://www.roboticsproceedings.org/rss13/p58.html.

[7] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine. "Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=S1vuO-bCW.

[8] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. "Reinforcement learning for robot soccer". In: *Autonomous Robots* 27 (2009), pp. 55–73.

[9] T. Westenbroek, F. Castaneda, A. Agrawal, S. Sastry, and K. Sreenath. "Lyapunov design for robust and efficient robotic reinforcement learning". In: *arXiv preprint arXiv:2208.06721* (2022).

[10] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. "Data efficient reinforcement learning for legged robots". In: *Conference on Robot Learning*. PMLR. 2020, pp. 1–10.

[11] A. Zhan, R. Zhao, L. Pinto, P. Abbeel, and M. Laskin. "A framework for efficient robotic manipulation". In: *Deep RL Workshop NeurIPS 2021*. 2021.

[12] Z. Hou, J. Fei, Y. Deng, and J. Xu. "Data-efficient hierarchical reinforcement learning for robotic assembly control applications". In: *IEEE Transactions on Industrial Electronics* 68.11 (2020), pp. 11565–11575.

[13] J. Tebbe, L. Krauch, Y. Gao, and A. Zell. "Sample-efficient reinforcement learning in robotic table tennis". In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 4171–4178.

[14] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. "Data-efficient deep reinforcement learning for dexterous manipulation". In: *arXiv preprint arXiv:1704.03073* (2017).

[15] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel. "Reinforcement learning on variable impedance controller for high-precision robotic assembly". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3080–3087.

[16] T. Z. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine. "Offline Meta-Reinforcement Learning for Industrial Insertion". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 6386–6393.

[17] Z. Hu, A. Rovinsky, J. Luo, V. Kumar, A. Gupta, and S. Levine. "REBOOT: Reuse Data for Bootstrapping Efficient Real-World Dexterous Manipulation". In: *arXiv preprint arXiv:2309.03322* (2024).

[18] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. "Residual Reinforcement Learning for Robot Control". In: *CoRR* abs/1812.03201 (2018). arXiv: 1812.03201. URL: http://arxiv.org/abs/1812.03201.

[19] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. Aparicio Ojea, E. Solowjow, and S. Levine. "Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5548–5555.

[20] I. Kostrikov, L. M. Smith, and S. Levine. "Demonstrating A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning". In: *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*. Ed. by K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu. 2023. URL: https://doi.org/10.15607/RSS.2023.XIX.056.

[21] J. Luo, P. Dong, Y. Zhai, Y. Ma, and S. Levine. "RLIF: Interactive Imitation Learning as Reinforcement Learning". In: *arXiv preprint arXiv:2311.12996* (2023).

[22] Y. Zhang, L. Ke, A. Deshpande, A. Gupta, and S. Srinivasa. *Cherry-Picking with Reinforcement Learning : Robust Dynamic Grasping in Unstable Conditions*. 2023. arXiv: 2303.05508 [cs.RO].

[23] T. Hester and P. Stone. "Texplore: real-time sample-efficient reinforcement learning for robots". In: *Machine learning* 90 (2013), pp. 385–429.

[24] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. "DayDreamer: World Models for Physical Robot Learning". In: *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*. Ed. by K. Liu, D. Kulic, and J. Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, 2022, pp. 2226–2240. URL: https://proceedings.mlr.press/v205/wu23c.html.

[25] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. "Deep Dynamics Models for Learning Dexterous Manipulation". In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by L. P. Kaelbling, D. Kragic, and K. Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1101–1112. URL: http://proceedings.mlr.press/v100/nagabandi20a.html.

[26] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn. "Offline Reinforcement Learning from Images with Latent Space Models". In: *Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control, L4DC 2021, 7-8 June 2021, Virtual Event, Switzerland*. Ed. by A. Jadbabaie, J. Lygeros, G. J. Pappas, P. A. Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger. Vol. 144. Proceedings of

Machine Learning Research. PMLR, 2021, pp. 1154–1168. URL: http://proceedings.mlr.press/v144/rafailov21a.html.

[27] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino. "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2062–2069.

[28] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. "Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost". In: *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. IEEE, 2019, pp. 3651–3657. URL: https://doi.org/10.1109/ICRA.2019.8794102.

[29] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. "Variational inverse control with events: A general framework for data-driven reward definition". In: *Advances in neural information processing systems* 31 (2018).

[30] K. Li, A. Gupta, A. Reddy, V. H. Pong, A. Zhou, J. Yu, and S. Levine. "MURAL: Meta-Learning Uncertainty-Aware Rewards for Outcome-Driven Reinforcement Learning". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 6346–6356. URL: http://proceedings.mlr.press/v139/li21g.html.

[31] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi. "Vision-language models as success detectors". In: *arXiv preprint arXiv:2303.07280* (2023).

[32] P. Mahmoudieh, D. Pathak, and T. Darrell. "Zero-Shot Reward Specification via Grounded Natural Language". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 14743–14752. URL: https://proceedings.mlr.press/v162/mahmoudieh22a.html.

[33] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D. Huang, Y. Zhu, and A. Anandkumar. "MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge". In: *NeurIPS*. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets%5C_and%5C_Benchmarks.html.

[34] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. "VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=YJ7o2wetJ2.

[35] Y. J. Ma, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman. "LIV: Language-Image Representations and Rewards for Robotic Control". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 23301–23320. URL: https://proceedings.mlr.press/v202/ma23b.html.

[36] A. Sharma, K. Xu, N. Sardana, A. Gupta, K. Hausman, S. Levine, and C. Finn. "Autonomous Reinforcement Learning: Benchmarking and Formalism". In: *arXiv preprint arXiv:2112.09605* (2021).

[37] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. "The Ingredients of Real World Robotic Reinforcement Learning". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rJe2syrtvS.

[38] A. Xie, F. Tajwar, A. Sharma, and C. Finn. "When to Ask for Help: Proactive Interventions in Autonomous Reinforcement Learning". In: *NeurIPS*. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/6bf82cc56a5fa0287c438baa8be65a70-Abstract-Conference.html.

[39] A. Sharma, A. M. Ahmed, R. Ahmad, and C. Finn. "Self-Improving Robots: End-to-End Autonomous Visuomotor Reinforcement Learning". In: *CoRR* abs/2303.01488 (2023). arXiv: 2303.01488. URL: https://doi.org/10.48550/arXiv.2303.01488.

[40] T. Seno and M. Imai. "d3rlpy: An Offline Deep Reinforcement Learning Library". In: *Journal of Machine Learning Research* 23.315 (2022), pp. 1–20. URL: http://jmlr.org/papers/v23/22-0017.html.

[41] A. Nair and V. Pong. "rlkit". In: *Github* (). URL: https://github.com/rail-berkeley/rlkit.

[42] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. *Stable Baselines*. https://github.com/hill-a/stable-baselines. 2018.

[43] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo. *TF-Agents: A library for Reinforcement Learning in TensorFlow*. https://github.com/tensorflow/agents. [Online; accessed 25-June-2019]. 2018. URL: https://github.com/tensorflow/agents.

[44] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. "Efficient online reinforcement learning with offline data". In: *arXiv preprint arXiv:2302.02948* (2023).

[45] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by L. P. Kaelbling, D. Kragic, and K. Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1094–1100. URL: http://proceedings.mlr.press/v100/yu20a.html.

[46] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. "RLBench: The Robot Learning Benchmark & Learning Environment". In: *IEEE Robotics Autom. Lett.* 5.2 (2020), pp. 3019–3026. URL: https://doi.org/10.1109/LRA.2020.2974707.

[47] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, P. P. Tehrani, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. *ORBIT: A Unified Simulation Framework for Interactive Robot Learning Environments*. 2023. eprint: arXiv:2301.04195.

[48] M. Ahn, H. Zhu, K. Hartikainen, H. Ponte, A. Gupta, S. Levine, and V. Kumar. "ROBEL: Robotics Benchmarks for Learning with Low-Cost Robots". In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by L. P. Kaelbling, D. Kragic, and K. Sugiura. Vol. 100. Proceedings of

Machine Learning Research. PMLR, 2019, pp. 1300–1313. URL: http://proceedings.mlr.press/v100/ahn20a.html.

[49] M. Heo, Y. Lee, D. Lee, and J. J. Lim. "FurnitureBench: Reproducible Real-World Benchmark for Long-Horizon Complex Manipulation". In: *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*. Ed. by K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu. 2023. URL: https://doi.org/10.15607/RSS.2023.XIX.041.

[50] S. Levine, C. Finn, T. Darrell, and P. Abbeel. "End-to-End Training of Deep Visuomotor Policies". In: *J. Mach. Learn. Res.* 17 (2016), 39:1–39:40. URL: http://jmlr.org/papers/v17/15-522.html.

[51] V. R. Konda and J. N. Tsitsiklis. "Actor-Critic Algorithms". In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by S. A. Solla, T. K. Leen, and K. Müller. The MIT Press, 1999, pp. 1008–1014. URL: http://papers.nips.cc/paper/1786-actor-critic-algorithms.

[52] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[53] S. Fujimoto, H. van Hoof, and D. Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by J. G. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1582–1591. URL: http://proceedings.mlr.press/v80/fujimoto18a.html.

[54] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018.

[55] A. Nair, M. Dalal, A. Gupta, and S. Levine. *Accelerating Online Reinforcement Learning with Offline Datasets*. 2020. arXiv: 2006.09359 [cs.LG].

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

[57] 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* 27 (2014).

[58] W. Han, S. Levine, and P. Abbeel. "Learning compound multi-step controllers under unknown dynamics". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 6435–6442.

[59] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[60] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz. *A Practical Approach to Insertion with Variable Socket Position Using Deep Reinforcement Learning*. 2018. arXiv: 1810.01531 [cs.RO].

[61] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik, N. Ye, S. Schaal, and J. Scholz. "Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study". In: *Proceedings of Robotics: Science and Systems*. Virtual, July 2021.

[62] O. Spector and D. D. Castro. *InsertionNet – A Scalable Solution for Insertion*. 2021. arXiv: 2104.14223 [cs.RO].

[63] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].