Enabling Robot Autonomy Through Real-World Practice

Laura Smith

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-108
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-108.html

May 16, 2025

Enabling Robot Autonomy Through Real-World Practice

By

Laura Michelle Smith

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair
Professor Pieter Abbeel
Professor Ken Goldberg
Professor Chelsea Finn

Spring 2025

Enabling Robot Autonomy Through Real-World Practice

Abstract

Enabling Robot Autonomy Through Real-World Practice

by

Laura Michelle Smith

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

Robots have demonstrated extraordinary capabilities over the past few decades, from performing surgeries to exploring space. Despite this progress, robots are not yet commonplace in our everyday lives; instead, they are confined to executing tasks where the humans behind them can account for everything the robot will encounter. The challenge in deploying robots that can be autonomous stems from the diversity and unpredictability of the physical world. Humans constantly encounter new situations, and we face this by quickly adapting to them as they arise. Could we also enable robots to face our unpredictable world by allowing them to learn online, from their real-world experiences? Reinforcement learning provides a framework for learning through interaction with and feedback from an environment. In this thesis, we study challenges in applying reinforcement learning to physical robot systems that are not confined to lab settings, and in doing so, propose algorithmic solutions, provide empirical analysis, and build practical training systems that demonstrate their efficacy. We begin by building a legged locomotion learning system that incorporates simulated pre-training, autonomous failure recovery, multi-task training, onboard sensors, and sample-efficient RL, and demonstrate that a small amount of real-world practice can enable effective fine-tuning in unstructured settings. We then show how to enable efficient learning with more complex reward functions, derived from supervision that is general and available in the real world: human preferences. We further simplify the assumptions and study learning *directly* in the real world, demonstrating a system capable of enabling a quadruped to learn to walk in various natural environments, purely from real-world experience. Lastly, we look towards learning more complex tasks by leveraging priors. First, we extend the efficient learning framework to effectively ingest offline, mixed-quality data. In discussing how this is practical for robotics applications, we show that this method enables flexible agile quadrupedal locomotion such as running jumps and bipedal walking. Finally, we explore how foundation models can adapt language-conditioned manipulation to new situations in the real world.

Contents

Contents i

List of Figures iv

1 Introduction 1
1.1 Publications . . . 4

I Enabling Training in the Wild 6

2 Legged Robots that Keep on Learning 7
2.1 Motivation . . . 7
2.2 Related Work . . . 9
2.3 Fine-Tuning Locomotion in the Real World . . . 11
2.4 System Design . . . 12
2.5 Experiments . . . 15
2.6 Discussion and Limitations . . . 19

3 Feedback-Efficient Interactive RL 21
3.1 Motivation . . . 21
3.2 Related Work . . . 23
3.3 Preliminaries . . . 24
3.4 PEBBLE . . . 25
3.5 Experiments . . . 28
3.6 Discussion . . . 34

II Learning in Real Time 35

4 Learning to Walk in 20 Minutes With Model-Free RL 36
4.1 Motivation . . . 36
4.2 Related Work . . . 37

4.3 Fast and Simple Real-World RL . . . 40
4.4 System Design . . . 42
4.5 Learning in the Real World . . . 44
4.6 Analysis of Design Decisions . . . 46
4.7 Conclusion . . . 48

5 Continuous Improvement with Real-World RL 50
5.1 Motivation . . . 50
5.2 Related Work . . . 52
5.3 System Design . . . 53
5.4 Efficient Learning of Legged Locomotion with Adaptive Policy Regularization 55
5.5 Real-World Results . . . 57
5.6 Simulated Analysis . . . 61
5.7 Conclusion . . . 64

III Leveraging Priors 65

6 Efficient Online RL with Offline Data 66
6.1 Motivation . . . 66
6.2 Related work . . . 68
6.3 Preliminaries . . . 69
6.4 Online RL with Offline Data . . . 69
6.5 Experiments . . . 75
6.6 Conclusion . . . 81

7 Learning and Adapting Agile Locomotion Skills by Transferring Experience 83
7.1 Motivation . . . 83
7.2 Related Work . . . 86
7.3 Preliminaries . . . 88
7.4 Transfer Learning for Agile Skills . . . 89
7.5 Learning Agile Locomotion Tasks with TWiRL . . . 91
7.6 Results . . . 96
7.7 Discussion and Future Work . . . 100

8 Flexible Robotic Manipulation via Dense Language Grounding 102
8.1 Motivation . . . 102
8.2 Related Work . . . 104
8.3 System Design . . . 105
8.4 Experiments . . . 108

8.5 Discussion 113

9 Conclusion **115**

10 Appendix **119**
10.1 State Entropy Estimator 119
10.2 Experimental Details 119
10.3 Effects of Sampling Schemes 121
10.4 Examples of Selected Queries 121
10.5 Detailed Experiments 126
10.6 Experimental Details 133

Bibliography **144**

List of Figures

2.1 **(Practicing locomotion skills in the real world with autonomous recovery)** We demonstrate real-world improvement through fine-tuning multiple skills to various real-world environments. The robot learns to walk back and forth on grass (topleft) and side-step on carpet (bottom left), while recovering seamlessly from failure (right). 7

2.2 **(Overview of fine-tuning system)** Example of our system. First, we pretrain skills (e.g., forward/backward pacing and reset) in simulation using RL. We then deploy the policies in the real world. The robot executes forward or backward pacing depending on which will bring it closer to the origin. After each episode, it automatically runs its reset policy. We continue to update the policies with real-world data using the same RL method for perpetual improvement. 10

2.3 **(Analysis of on-board state estimation quality)** Real-world state estimation compared to motion capture for a robot walking indoors. Yaw and yaw velocity are very accurate, linear velocity is acceptable, and x- and y-position drift over the course of multiple episodes. 13

2.4 **Recovery controller training conditions** Examples from the initial state distribution used to train the reset policy in simulation. To get these states, we drop the robot from about a half meter above the ground with a random initial orientation of the robot's torso. Specifically, its roll, pitch and yaw are drawn from uniform distributions over the intervals $[-\frac{3\pi}{4}, \frac{3\pi}{4}], [-\frac{\pi}{4}, \frac{\pi}{4}]$, and $[-\pi, \pi]$, respectively. 14

2.5 **(Simulation fine-tuning analysis)** We report each adapted policy's performance (mean and standard deviation across 10 trials) in the target domain (pictured in the top row) with respect to the samples used for adaptation. Prior methods that learn adaptation strategies during training (pink and yellow) excel in environments that are similar to those seen during training (left) but fail in environments that are sufficiently different (middle, right). In contrast, our fine-tuning method (blue) continues to improve under all circumstances. We also note that our use of REDQ (blue) improves over SAC (purple). 16

2.6 **(Real-world fine-tuning results)** Top: Learning curves for all the real-world fine-tuning experiments showing the average return of data collected by a stochastic policy during each iteration of training. Bottom: We evaluate each policy with a deterministic policy before and after fine-tuning and report mean and standard deviation over ten trials. In all domains, fine-tuning leads to improvement, and the improvement is particularly pronounced on the most difficult surfaces, such as the lawn and memory foam mattress. See the project website for videos of each of these policies throughout training. 18

2.7 **(Example rollouts)** 1st and 2nd Rows: Example rollouts of the pacing policies before and after fine-tuning. The figure depicts a timelapse of a rollout of each policy, where the opacity of the robot indicates time progression (i.e., more opaque corresponds to more recent). Before fine-tuning, the forward policy struggles to make progress, while the pre-trained backwards policy makes fast progress but is quite unstable. After fine-tuning, both policies make forward progress without falling. 3rd to 5th Rows: Example rollouts of the side-step policies before and after fine-tuning. On all terrains, the pre-trained policies fail to complete the task without falling. After fine-tuning, the policies reliably execute the skill on all three domains. 20

3.1 **(Overview of PEBBLE)** First, the agent engages in unsupervised pre-training during which it is encouraged to visit a diverse set of states so its queries can provide more meaningful signal than on randomly collected experience (left). Then, a teacher provides preferences between two clips of behavior, and we learn a reward model based on them. The agent is updated to maximize the expected return under the model. We also relabel all its past experiences with this model to maximize their utilization to update the policy (right). 22

3.2 **(PEBBLE evaluation environments)** Examples from the environments we test on. We consider learning a variety of complex locomotion and manipulation skills through interacting with a scripted or human trainer. 29

3.3 **(PEBBLE locomotion results)** Learning curves on locomotion tasks as measured on the ground truth reward. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and Preference PPO is indicated by dotted lines of the corresponding color. 29

3.4 **(PEBBLE manipulation results)** Learning curves on robotic manipulation tasks as measured on the success rate. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and Preference PPO is indicated by dotted lines of the corresponding color. 30

3.5 **(Ablation study on Quadruped-walk)** (a) Contribution of each technique in PEBBLE, i.e., relabeling the replay buffer (relabel) and unsupervised pre-training (pre-train). (b) Effects of sampling schemes to select queries. (c) PEBBLE with varying the length of the segment. The results show the mean and standard deviation averaged over ten runs. 31
3.6 **(Results using real human feedback)** Novel behaviors trained using feedback from human trainers. The corresponding videos and examples of selected queries are available in the supplementary material. 31
3.7 **(Comparing human to simulated feedback)** Five frames from agents trained with (a) human preference and (b) hand-engineered reward from DM-Control benchmark. 31

4.1 **(Overview of demonstration)** We demonstrate that deep reinforcement learning can be used to efficiently train a quadruped robot directly on various real world terrains, e.g., flat ground (blue); soft, irregular mulch (green); grass (red); and a hiking trail (yellow), acquiring effective gaits within 20 minutes of training. 36
4.2 **(Real-world test terrains)** Examples of learned gaits acquired on a variety of real-world terrains. Left to right: flat, solid ground covered in dense foam mats; a 5cm memory foam mattress; loose ground comprised of eucalyptus bark; a grassy lawn; a gently sloped hiking trail. 44
4.3 **(Real-world learning from scratch results)** Learning curves for all the real-world experiments showing the average velocity of the stochastic policy with respect to real-world, wall-clock time. Note that the robot runs continuously, training in between sending actions to the physical hardware. 1000 time-steps corresponds roughly to 1 minute of training time, all things considered. 46
4.4 **(Empirical analysis of design choices in simulation)** Experimental evaluation of: (a) performance for different values of the damping parameter for the position PD controller; (b) ablations of various task setup choices; (c) the effect of the frequency of policy updates (between time-steps versus episodes); (d) regularization and normalization methods for efficient RL. Each curve and shaded region represents the average and standard deviation across 10 random seeds. 48

5.1 **(Overview of APRL)** APRL uses a novel action space regularization technique based on dynamics prediction error to modulate exploration over the course of training. This enables real-world quadrupedal learning that can traverse challenging terrains and continually adapt to changes in dynamics. . . . 51

5.2 **(Depiction of APRL system)** We train the robot with a flexible constraint, represented by blue semicircles around the joints. It collects experience, storing it in a replay buffer for training an actor and critic, as explained in Section 5.3, alongside a predictive dynamics model. The model's prediction error adjusts the constraint's bounds, either tightening (for high error) or relaxing (for accurate predictions). This adjustment is incorporated into the actor's loss, as specified in Equation 5.1. 56
5.3 **(Test environment visualization)** We visualize the different environments we test in: Grass, Ramp, Mattress, and Frozen Joint. We indicate the start and goal locations for the situations in which there is a path for the robot to traverse and is evaluated on its time to finish the entire path. 58
5.4 **(Qualitative comparison of policies)** We compare the gaits learned, (top) Restricted and (bottom) APRL, from scratch on flat ground by showing a time-lapse of the policies rolled out for 5 seconds each. Our policy learned to use its front legs to step and propel its back legs in a cantering-like manner whereas the Restricted policy drags and slides across the ground. 59
5.5 **(Qualitative comparison on new terrain)** We show a 5 second time-lapse of evaluating different policies on the mattress. The Restricted method tries to slide on the mattress, which slows it down significantly. APRL policies have a higher foot clearance, so they are able to traverse it more efficiently and, after fine-tuning, with fewer falls. 60
5.6 **(Forward velocity achieved during training)** The Restricted method learns more efficiently at first but is unable to keep improving. Meanwhile, our method still learns to walk quickly but acquires a maximum velocity of 0.62 m/s. Note that the dip at 40k steps is due to parameter resets to improve plasticity (line 19 in Algorithm 5). 60
5.7 **(Real-world velocity comparisons)** In all scenarios except frozen joint, APRL significantly outperforms the Restricted method in terms of its velocity when tested in new scenarios. With just minutes of fine-tuning, APRL significantly improves performance in all settings except on the ramp, where it is comparable. 61
5.8 **(Real-world finish time and fall count)** (Left) We report the time taken to traverse a path relative to the Restricted method and absolute fall count. On Ramp and Grass, APRL is **2x** faster, and on Mattress, APRL is **almost 4x** faster. 62
5.9 **(Simulation comparison results)** We report each policy's performance measured by the falls, average velocity, and return (mean and standard error across 5 seeds) with respect to the number of time steps. We find that APRL is the only method that effectively balances achieving high velocity while regulating the number of falls such that it is feasible to run in the real world. 63

5.10 **(Simulation ablation results)** We compare to versions of APRL that use (a) a hard constraint (b) non-adaptive regularization (c) regularization via the reward function. These either have too many falls or do not progress on *forward velocity*, showing the importance of all design components of APRL. . 64

6.1 **(Overview of RLPD results)** Our approach, RLPD, extends standard off-policy RL and achieves reliable state-of-the-art online performance on several tasks using offline data. Here we show the difficult D4RL AntMaze domain (10 seeds, 1 std. shaded), averaged over all 6 tasks. 66
6.2 **(LayerNorm Ablation)** Using SAC with our symmetric sampling method can result in instabilities due to diverging Q-values; with LayerNorm in the critic this disappears, improving performance. 70
6.3 **(Didactic example of extrapolation with and without LayerNorm)** We fit data (left) with a two-layer MLP without LayerNorm (center) and with LayerNorm (right). LayerNorm bounds the values and prevents catastrophic overestimation. 72
6.4 **(Results per domain)** RLPD exceeds prior state-of-the-art performance on a number of different popular benchmarks whilst being significantly simpler. Results are aggregated over **21 different environments** (10 Seeds, 1 std. shaded). In each case, we compare to the prior best known work (IQL + Finetuning in Adroit and AntMaze, Off2On in Locomotion), and SACfD, a canonical off-policy approach using offline data. 75
6.6 **(Results on vision-based tasks)** Our approach generalizes to vision-based domains, providing consistent improvements over existing approaches. 77
6.5 **(Effects of changing UTD with vision-based task)** Increasing UTD with RLPD greatly improves sample efficiency from pixels. 77
6.7 **(LayerNorm ablation)** LayerNorm is crucial for strong performance, particularly when data are limited or narrowly distributed. 78
6.8 **(Empirical evidence for proposed workflow on challenging tasks)** Our recommended starting design choices and workflow leads to strong performance on all tasks. 79
6.10 **(Buffer sampling analysis)** Comparing the effects of design decisions in sampling from the offline data. 80
6.9 **(Critic regularization analysis)** In general, critic ensembling provides the best performance. Dropout performs worse in sparse reward tasks. 80
6.11 **(Effect of replay proportion)** RLPD is not sensitive to replay proportion; 50% offers the best compromise between variance, speed of convergence, and asymptotic performance. 81

7.1 **(Agile skills learned with TWiRL)** TWiRL enables the A1 robot to jump repeatedly (left) and walk to a goal location on its hind legs (right). 83

7.2 **(Overview of practical applications)** We identify that several fundamental challenges in learning agile locomotion skills can be ameliorated by casting them as transfer learning problems, then applying a simple, generic method that involves a simple modification to off-the-shelf off-policy algorithms. We show that our framework is versatile—with the same method, we can **(top)** generalize a policy trained to track a reference motion of a jumping dog to learn to jump over randomly placed obstacles; **(middle)** take a policy that is trained to kick up onto the robot's hind legs to then use bipedal locomotion to navigate to randomly sampled goals; and **(bottom)** enable efficient fine-tuning in new environments. 84
7.3 **(Comparing jumping skills learned with and without prior data)** Examples of our policy (outlined in yellow), which incorporates both online training and data from a motion imitation policy, compared to two policies (outlined in blue) trained from scratch with the same reward function. While naïvely optimizing for the task either exploits the simulator to learn an unnatural motion (middle) or fails completely (bottom), the policy trained by incorporating prior data exhibits a graceful jump. 93
7.4 **(Comparing bipedal skills learned with and without prior data)** Examples of our policy (outlined in yellow), trained with data from a robot that can already stand on its hind legs, compared to a baseline policy (blue) trained from scratch. Without this added bias, the baseline policy learns to scoot toward the goal on its knees. Our policy gracefully kicks up to standing and navigates to the goal on 2 legs. 94
7.5 **(Simulated fine-tuning environments)** Illustrating the diverse environments to which we adapt the source policy (trained in the environment labeled 'none' to indicate no modification). In clockwise order: the default, non-randomized environment; a sloped terrain; bumpy terrain; a low-gravity environment; a stochastic environment simulating motor weakening; a simulated ice rink. 95
7.6 Learning curves for the jumping (left) and bipedal navigation (right) tasks comparing learning from scratch (solid yellow curve) to TWiRL (solid blue curve) using data from a source policy (dotted green line), shown to 1M steps. Since the policies from scratch had not yet converged for the jumping task, we gave an additional 2 million steps and visualized the average at convergence as well (yellow dotted line); we omit its variance for clarity. We see that in both cases TWiRL is able to far surpass the policies learned from scratch and the source policies. 97

7.7 **(Real-world jumping rollouts)** Example successful rollouts from evaluating our jumping policy in the real world with different spacings of hurdles. From top to bottom the spacing is at 2.4m, 2.8m, and 3.6m, with success rates over 8 trials of **75%, 100%, and 75%**, respectively. We see that our policy exhibits the desired behavior (derived from the motion imitation policy) while being robust to task variation, but furthermore, it is able to be deployed on hardware. Videos of all evaluation trials can be found at `https://sites.google.com/berkeley.edu/twirl`. 98

7.8 **(Real-world bipedal rollouts)** Comparison of our bipedal navigation policy to the baseline trained from scratch evaluated in the real world. Over 5 trials, our policy spent an average of 12 seconds upright and successfully approached the goal, whereas the baseline consistently scooted on its knees and never stood upright. 99

7.9 **(Results adapting to different dynamics)** Evaluating TWiRL in adapting the jumping and bipedal navigation skills to 5 different environments (see Section 7.5) given a budget of 100k samples—corresponding to roughly 1.5 hours real-world time without any overhead. On the left, we show the learning curves for two environments in order to visualize the learning speed/dynamics and on the right, we summarize results for all environments. We see that in all cases, a variant of TWiRL is able to successfully allow the policies to adapt to new environments. 100

8.1 **(System diagram for STEER)** At training time, we re-annotate an offline dataset of diverse robot behaviors at training time, focusing on describing the primitive skills used to manipulate objects and, specifically, on annotating *how* the robot performed each skill. We then use this re-annotated dataset to train a language-conditioned low-level policy (RT-1 in our case). At inference time, when given a complex instruction like "pick up the flower pot *without disturbing the plant*", a high-level system (VLM or human) identifies the appropriate low-level skills and determines how to perform them. This emphasis on the "how" enables more contextual behavior. 102

8.2 **(Visualization of grasp angle labels)** Anchor vectors and their semantic labels. Purple, green, and pink vectors represent side, top-down, and diagonal grasps. 107

8.3 **(Real-world results)** Results on grasping in unseen scenarios and performing a new task, with human or VLM guidance. We find that by having access to and being able to reason about extracted low-level strategies enables higher success in OOD scenarios than the baseline RT-1 model and a state-of-the-art VLA. 108

8.4 **(Sample initial conditions)** for the new object-grasping scenarios evaluated. (top left) a kettle with a handle extending above it, (top right) a potted plant, (bottom) 2/15 scenes for `Fruit in Clutter`. The kettle should be grasped over top. In order to avoid disturbing the plant, the flower pot should be grasped around its body. Lastly, the fruits should be grasped while avoiding knocking over the other objects in the scene. 109
8.5 **(Grasp steerability)** of OpenVLA, RT-1, and STEER. We test the steerability to grasp an object in different ways that would be appropriate for different, unseen tasks, e.g., in order to pour out of the Coke can, the robot should grasp the can around its body. When prompted to "Grasp the Coke can" from the top (top row) versus the side (bottom row), models without dense annotation show no perceivable change, while our densely labeled model adjusts its behavior, enabling new downstream tasks. 110
8.6 **(STEER solving a new complex manipulation task leveraging steerable low-level skills)** 113

10.1 **(Locomotion sampling ablations)** Learning curves of PEBBLE with 1400 pieces of feedback by varying sampling schemes. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. 121
10.2 **(Meta-world sampling ablations)** Learning curves of PEBBLE with various sampling schemes on the Meta-world tasks. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. 122
10.3 **(Examples from the selected queries to teach the Cart agent)** 123
10.4 **(Examples from the selected queries to teach the Quadruped agent)** 124
10.5 **(Examples from the selected queries to teach the Hopper agent)** . . . 125
10.6 **(Full Adroit results)** 126
10.7 **(Full AntMaze results)** 127
10.8 **(SAC with and without Layer Normalization)** 128
10.9 **(RLPD v.s. SAC + LN + Offline Data)** 129
10.10 **(RLPD v.s. IQL + Finetuning)** We show offline pre-training steps for IQL using negative indices. 130
10.11 **(D4RL Ablations)** 131
10.12 **(D4RL Ablation on Expert Data)** 131
10.13 **(VD4RL Ablations)** 132
10.14 **(Experimental environment visualization)** 133
10.15 **(Distribution of training goals)** (Left) We visualize the bipedal navigation task, where each colored dot is a sample from the goal distribution we defined. The small dotted circle corresponds to a radius of 1m and the larger circle corresponds to a radius of 2m. 137
10.16 **(Full dynamics adaptation results)** Individual learning curves for every task and target environment combination. 141

10.17**(Effect of using data from π_{src})** 142
10.18**(Effects of using high UTD algorithms for incorporating off-policy data)** We find that using an UTD ratio > 1 is important for stable learning particularly when learning the jumping policies in a new environment. 142

Acknowledgements

I am deeply grateful to my advisor, Sergey Levine, who is one of the most insightful, thoughtful, and kind people I have had the privilege to learn from. His support throughout my career has been unwavering, and I thank him for encouraging me to pursue a bold research agenda and being there every step of the way. My journey into AI research began as an undergraduate at Berkeley in 2018, where I had a profoundly positive experience working with Pieter Abbeel and Marvin Zhang. Pieter and Marvin were my first research mentors and played pivotal roles in shaping my path. I cannot thank them enough for their incredible patience, generosity, and mentorship. In addition to Sergey and Pieter, I would also like to warmly thank Professors Ken Goldberg and Chelsea Finn for supporting me throughout the years and helping to improve this thesis.

During my PhD, I have been extremely fortunate to work with and learn from many incredible researchers. I would especially like to thank Jason Peng, Kimin Lee, and Ilya Kostrikov for investing so much time and energy in encouraging, teaching, and guiding me as a junior PhD student. I also had the pleasure of working closely with colleagues across multiple institutions, so much so that many of us ended up writing multiple papers together over the years! Thank you all for the years of wonderful collaborations: Chelsea, Ilya, Jason, Kimin, Pieter, Annie Chen, Chase Kew, Jie Tan, Philip Ball, Sehoon Ha, Tianyu Li, Yunhao Cao, and Zhongyu Li. In addition to research at Berkeley, I was lucky to experience research in industry labs. First, learning about large-scale model training at Google DeepMind with guidance from Alex Irpan and Ted Xiao. Then deploying robot foundation models out of the lab with Chelsea and the team at Physical Intelligence. I am so grateful to my hosts and collaborators, who graciously afforded me these formative opportunities that have deeply influenced my outlook on open-world robot deployment.

I started graduate school under unique circumstances – during the pandemic, in lockdown in my childhood bedroom. I am so grateful to RAIL, RLL, and the broader BAIR community for fostering a positive and supportive environment. To my cohort that started with me in these unusual times — Ameesh Shah, Dave Epstein, Dibya Ghosh, Fangchen Liu, Katie Kang, Qiyang Li — I could not have asked for a better group to navigate this journey alongside. While they were not in my cohort at Berkeley, I am thankful to have been friends with Russell Mendonca and Jesse Zhang, who shared much of our first year virtually and continue to be such important influences in my life and research. I am especially thankful to senior students who were always willing to dedicate their time to listen and offer their wisdom — Kelvin Xu, Michael Chang, Michael Janner, Vitchyr Pong, Dhruv Shah, and Annie Xie. I also thank the junior students, especially Seohong Park, Kevin Black, Homer Walke, William Chen, Kyle Stachowicz, and Catherine Glossop, for reminding me why I chose to pursue a PhD with their passion and enthusiasm for research. I also want to thank my two best friends who I met due to the program,

Kevin Zakka and Philipp Wu, for teaching me so much about robotics through working alongside each other every day, sharing our hobbies, and generally bringing me so much joy throughout this process.

To Berkeley, my home for almost 10 years now, I am so grateful for the community that enriched my life outside of research: the lovely people at CorePower Yoga and the Y; Artis, Coro, Yali's, and Peet's on Vine; Polished Nail Spa. Last, but certainly not least, I thank my family for their unconditional love and support. To my brother David, for always setting a great example and high bar to strive for. And to my parents, Ok and Michael Smith, who have given me everything and more than I could have ever asked for.

Chapter 1

Introduction

Humans must constantly contend with an immense amount of complexity presented by our world, from mundane variations in everyday routines—like having to take a detour due to road construction—to more drastic ones like traveling to a country where people drive on the other side of the road or picking up a new language. The demands on our decision-making and control processes are highly dynamic, as not only our environment, but also our own embodiments and objectives are ever-changing. If we desire robots capable of being deployed in our world, acting with human or even animal-like proficiency, they must also contend with these challenges. Enabling this autonomy in open-world environments remains a formidable challenge. Indeed, robots have demonstrated striking physical capabilities—such as being used in life-saving surgeries, extraterrestrial exploration, and many more real-world applications—yet they have not become ubiquitous fixtures in everyday life.

We navigate these complex dynamics adeptly because we can adapt to new situations, learn new concepts and skills, and generally grow more competent over time from these gathered experiences. In contrast, robots have largely been designed with programs that do not adapt or improve with more experience. To be deployed in an open-ended manner, that is, without controlling the environments they can be placed in or the particular tasks they will need to perform, robots must be designed to face unseen scenarios that they will inevitably encounter in the real world. This requirement for powerful generalization has led many roboticists to explore data-driven solutions in favor of hand-engineered ones. Imitation learning is one popular data-driven approach that has gained popularity, especially in real-world applications where large-scale human data collection is viable such as autonomous driving [1–4], due to its effectiveness in distilling expert demonstrations into learned policies that can then be deployed autonomously [5–11]. While these works provide an excellent framework to mimic human-like behaviors, real-world data collection that is bottlenecked by humans is limiting as it is costly and difficult to scale in diversity due to physical constraints. Another data-driven approach that has proven highly

effective in automatically producing extremely robust controllers is using reinforcement learning in simulation, especially in settings where it is viable to procedurally generate environments that reflect the kinds of environments the robot will be expected to perform in the real world [12–20]. However, it is difficult to predict all the possible situations a robot will encounter, including environmental conditions and tasks, during deployment and enumerate them at training time. In either case, humans need to curate the training conditions and can only hope that the resulting policy generalizes after learning from a fixed dataset.

We propose a vision for robots that face our unpredictable world by endowing them with the ability to learn from their real-world experiences. A data-driven approach wherein robots themselves can *autonomously* collect and then learn from their own data would, in principle, allow robots to 'generalize' by being able to adapt to the demands of their environment. The central question of this thesis is: *how do we build physical systems with the ability to learn autonomously, from their own lived experiences?* Reinforcement learning (RL) provides a conceptually elegant solution, offering a framework in which agents learn by exploring in an environment and updating their behavior based on feedback they receive during that interaction. However, the practical application of RL algorithms on physical hardware in the real world is not straightforward. To exemplify this, the major successes in reinforcement learning have been in solving problems where the exploration component is inexpensive, and getting feedback from the environment is straightforward. For example, RL has been used to discover superhuman strategies in video and board games [21–24]. The purpose of the chapters to follow is to study aspects of learning under conditions that reflect those we want robots to eventually be able to learn in. The organization of this document is as follows:

I. Enabling training in the wild

RL for real robots has often been confined to relatively sterile environments, where we can carefully control and measure things, and where humans can be there to scaffold the learning process [25–27]. But if we hope to scale to the open world, systems must be largely autonomous, i.e., they must not rely heavily on human intervention or heavily instrumented environments. In this section, we discuss steps towards building such scalable systems.

i. In Chapter 2, we start with building a robot learning system 'in the wild'. In particular, we take a legged robot, one that can traverse and face lots of natural diversity, into the outdoors. Most work in learned locomotion focuses on zero-shot generalization from simulation; however, this approach is fundamentally limited by what the robot can learn in the simulator [28–34]. We first explored using real-world data to fine-tune policies trained in simulation, building an autonomous system using interleaved forward and backward pacing

training and automated resets. This endeavor revealed that it is possible to improve dynamic locomotion skills in noisy, unstructured environments. Perhaps surprisingly, the largest bottleneck was in facilitating the training process, reducing the human effort in scaffolding the training process.

ii. For legged locomotion, providing reward supervision in the real world is challenging, but in Chapter 2 we detail how to make it practical with only on-board sensors for various agile locomotion skills. However, supervising based on only intrinsic features (i.e. root linear velocity and joint poses) is not sufficient in general, especially for manipulation tasks. In Chapter 3 we explore how to use sparse human feedback to provide supervision for learning complex behaviors using the framework of preferences [35]. The focus of this chapter is to make this more practical for real-world training by reducing the burden on human supervision — by showing how to make it possible with the same sample-efficient RL algorithm and how to sparingly use human queries.

II. Learning in real time

Exploration for physical robots is extremely expensive: robots incur wear and tear, they can damage the environment around them, and the human supervision requirement has been prohibitively expensive. In Chapter 2, the most burdensome aspects of exploration where behavior needs to be discovered and the behavior is most unsafe are thus offloaded in simulation pre-training. However, there are many things that are incredibly difficult to simulate (e.g. dynamics of deformable objects) and humans learn many things directly on-site. Here we explore a radical but conceptually elegant proposition — end-to-end learning, directly in the real world, obviating the need for simulation, demonstrations, or other crutches.

i. In Chapter 4, we build on the system that facilitated real-world training from Chapter 2. Here, we lift another fundamental requirement of pre-training in simulation and push the limits in terms of efficiency. We demonstrate through careful analysis of algorithmic and system design decisions that a high-dimensional legged robot can in fact learn a completely new skill in various 'in the wild' situations, completely on its own in just a handful of minutes, without any simulation.

ii. Next, we aim for continued improvement by addressing challenges in exploration and continual learning. That is, in Chapter 4, we defined manual guardrails to make exploring from scratch feasible in the real world but comes at a cost in performance. In Chapter 5, we extended this work to be able to learn increasingly complex behavior by modulating the robot's exploration according to its familiarity, i.e., we design an automatic curriculum to ensure that the robot does not fall excessively while also gradually increasing its limits as it becomes more competent.

III. Leveraging priors

The previous chapters have demonstrated that robots can learn drastically new behaviors remarkably quickly. However, there is a limit to how much a robot can learn from first-hand experience alone. In this section, we will discuss mechanisms for leveraging other sources of knowledge to bootstrap learning of more complex skills.

i. In Chapter 6, we aim to augment the same training algorithm as used in all previous chapters with other *data* sources. That is, can we learn *more* efficiently if we incorporate offline data? This chapter includes extensive experimental analysis for an incredibly simple and effective recipe we refer to as RLPD.

ii. How does leveraging offline data for online learning translate to better robot capabilities? Data we have access to for real robots is likely fairly narrow or highly suboptimal with respect to a new skill. In Chapter 7, we aim to transfer knowledge from existing controllers (e.g. MPC, motion imitation, undirected behavior) to bootstrap learning more complex ones (e.g. adaptive, robustness, goal-conditioned behaviors). A crucial requirement for the learning algorithm, then, is that it must be agnostic to the quality or relevance of source controllers. RLPD effectively transfers knowledge through suboptimal *data* during online training of the new task and, as a result, we demonstrated highly agile, running jumps and bipedal walking on a real quadruped.

iii. In Chapter 8, we investigate how we can leverage foundation models to imbue robots with commonsense reasoning as they approach new situations. We show that with sufficiently dense language grounding, robots can learn to perform new tasks by lifting the action space into a semantically meaningful abstraction layer (i.e. language) that vision-language models natively reason in.

Finally, we conclude by discussing some open challenges and perspectives on promising future directions.

1.1 Publications

This thesis comprises research conducted in collaboration with many colleagues; the following is a list of the publications that support each chapter:

Chapter 2: Laura Smith, J. Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, Sergey Levine. Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World. ICRA 2022 [36]

Chapter 3: Kimin Lee*, Laura Smith*, Pieter Abbeel. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training. ICML 2021 [37]

Chapter 4: Laura Smith*, Ilya Kostrikov*, Sergey Levine. Demonstrating a Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning. RSS Demo 2023 [38]

Chapter 5: Laura Smith*, Yunhao Cao*, Sergey Levine. Grow Your Limits: Continuous Improvement with Real-World RL for Robotic Locomotion. ICRA 2024 [39]

Chapter 6: Philip J. Ball*, Laura Smith*, Ilya Kostrikov*, Sergey Levine. Efficient Online Reinforcement Learning with Offline Data. ICML 2023 [40]

Chapter 7: Laura Smith, J. Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, Jie Tan, Sergey Levine. Learning and Adapting Agile Locomotion Skills by Transferring Experience. RSS 2024 [41]

Chapter 8: Laura Smith, Alex Irpan, Montserrat Gonzalez Arenas, Sean Kirmani, Dmitry Kalashnikov, Dhruv Shah, Ted Xiao. STEER: Flexible Robotic Manipulation via Dense Language Grounding. ICRA 2025 [42]

Part I

Enabling Training in the Wild

Chapter 2

Legged Robots that Keep on Learning

Figure 2.1: **(Practicing locomotion skills in the real world with autonomous recovery)** We demonstrate real-world improvement through fine-tuning multiple skills to various real-world environments. The robot learns to walk back and forth on grass (topleft) and side-step on carpet (bottom left), while recovering seamlessly from failure (right).

2.1 Motivation

Legged robots possess a unique physical capability to traverse a wide range of environments and terrains, from subterranean rubble to snowy hills [32, 43]. However, fully realizing this capability requires controllers that can effectively handle this broad range of environments. Engineering such robust controllers for each robot is a labor-intensive process, requiring human expertise and precise modeling of the system dynamics [44–46]. Reinforcement learning (RL) algorithms have been used to automatically learn robotic locomotion skills in a wide range of contexts, both in simulation and in the real world [28, 31, 47–56]. However, in order for these methods to handle the full range of

environments that the robot will encounter at test-time, they must be trained in an appropriately broad range of conditions – in a sense, the RL approach exchanges the burden of controller engineering for the burden of training-time environment engineering. While much of the work on learning locomotion skills has focused on training robust skills that can generalize to a variety of test-time conditions (e.g., different terrains) [34, 57], they all share the same fundamental limitation: they lack any recourse when the test-time conditions are so different that the trained controllers fail to generalize. Here we are specifically interested in the case where perfect zero-shot generalization is impossible. In this case, what can the robot do? In this chapter, we explore the approach of *real-world fine-tuning*: when the robot inevitably fails, it would need the mechanisms necessary to recover and *fine-tune* its skills to this new environment.

Although in principle RL provides precisely the toolkit needed for this type of adaptation, in practice this kind of fine-tuning presents a number of major challenges: the fine-tuning must be performed rapidly, under real-world conditions, without reliance on external state estimation or human assistance. The goal is to design a complete system for fine-tuning robotic locomotion policies under such real-world conditions. In our proposed framework, the robot would first attempt the desired locomotion task in some new environment, such as the park shown in Figure 2.1. Initially, it may fall because the uneven ground is not compatible with its learned policy. At this point, it should immediately stand back up using an agile learned reset controller, make a few more attempts at the task, and then use the collected experience to update its policy. To enable open world learning, the reward signal for RL must be obtained from the robot's own onboard sensors, and the robot must keep attempting the task until it succeeds, improving with each trial. This process must be successful both where it generalizes well (and is usually successful) and where it generalizes poorly (and fails on most initial trials). Concretely, we utilize motion imitation [57] to provide a general recipe for learning agile behaviors. To ensure that the robot operates autonomously, we use a learned recovery policy that enables the robot to quickly and robustly recover from falls. This autonomous process can either fine-tune one skill at a time, or fine-tune multiple complementary skills together, such as a forward and a backward walking motion. Due to our choice of RL formulation, learning these additional policies is a simple, straightforward extension. For efficient and stable real-world training, we opt for an off-policy RL algorithm that uses randomization over ensembles to stabilize and substantially improve the sample-efficiency of Q-learning methods [58].

The main contribution presented in this chapter is a system for real-world autonomous fine-tuning of agile quadrupedal locomotion skills. To our knowledge, our system is the first to show real-world fine-tuning using RL, with automated resets and onboard state estimation, for multiple agile behaviors with an underactuated robot. In our experiments, we take advantage of simulation data to pre-train a policy, reaping the safety and efficiency benefits of training in simulation, while retaining the ability to continue learning

in new environments with real-world training. Although the particular components we integrate into our real-world fine-tuning system are based on prior works, the combination of these components is unique to our system and together, they enable efficient real-world fine-tuning of agile and varied locomotion behaviors, together with highly efficient resets between trials and recoveries from falls. We demonstrate in our experiments that our system enables an A1 quadruped robot to learn dynamic skills, such as pacing forwards and backwards in an outdoor grass field, and side-stepping on 3 indoor terrains: carpet, doormat with crevices, and memory foam.

2.2 Related Work

Robotic locomotion controllers are typically constructed via a combination of footstep planning, trajectory optimization, and model-predictive control (MPC) [59]. This has enabled a range of desirable gaits, including robust walking [46] and high-speed bounding [60]. However, such methods require characterization of the robot's dynamics and typically a considerable amount of manual design for each robot and each behavior. RL provides an appealing approach to instead learn such skills, both in simulation [53–55] and in the real world [28, 31, 47, 48, 50, 56]. Due to safety considerations and the data intensive nature of RL algorithms, RL-based locomotion controllers are often trained in simulation. Various methods are used to improve transfer to the real world, such as building high fidelity simulators [28, 61], using real world data to improve the accuracy of simulators [62–64], and simulating diverse conditions to capture the variations a robot may encounter during real world deployment [14, 32, 65]. However, legged robots are capable to traverse such a wide variety of terrains. It is thus difficult to anticipate all the conditions they may encounter at test-time, and even the most robust learned policies may not generalize to every such situation.

Another line of work trains adaptive policies by incorporating various domain adaptation techniques to perform few-shot adaptation [29, 31, 57, 61, 66, 67]. Particularly related to our work, two prior works have proposed to train locomotion policies in simulation [34, 57] that include a learned adaptation structure, which infers a latent or explicit descriptor of the environment. However, although such policies are adaptive, their ability to adapt is also limited by the variability of conditions seen at training-time — if the test-time conditions differ in ways that the designer of the simulation did not anticipate, they may likewise fail, as we illustrate in our experimental comparison in Subsection 2.5.1. Thus, in this work, we rather aim to perform consistent adaptation through fine-tuning with RL, and present a method that enables continuous improvement under any test-time condition in the real world. Julian et al. [68] uses an off-policy model-free RL approach to fine-tune a visual grasping policy to a variety of conditions that are not covered during pre-training. Our approach similarly uses off-policy model-free learning to continuously learn subject to changes in the environment, but we instead consider a variety of skills and challenges

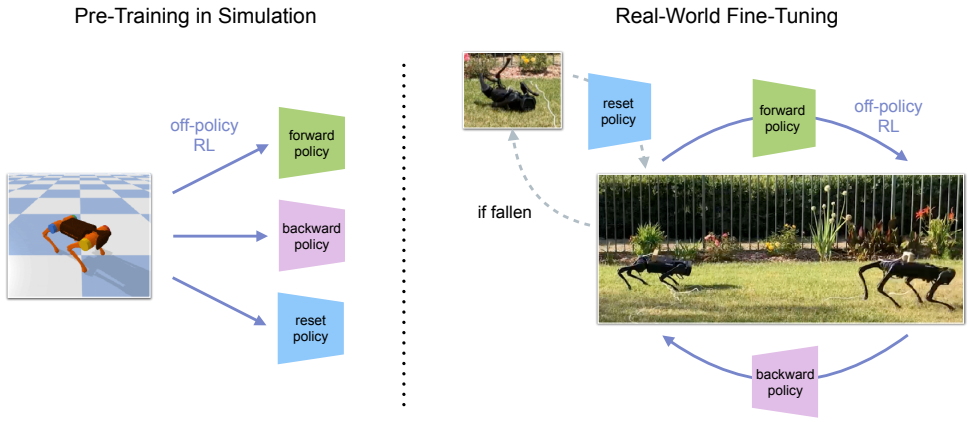


Figure 2.2: **(Overview of fine-tuning system)** Example of our system. First, we pre-train skills (e.g., forward/backward pacing and reset) in simulation using RL. We then deploy the policies in the real world. The robot executes forward or backward pacing depending on which will bring it closer to the origin. After each episode, it automatically runs its reset policy. We continue to update the policies with real-world data using the same RL method for perpetual improvement.

introduced by learning legged locomotion skills, such as underactuation and falling.

Several works have approached the challenge of training locomotion policies in the real world. This approach, however, has yet to scale to more complex motions due to the supervision requirements: these systems often rely on heavy instrumentation of the environment, such as motion capture systems, to provide reward supervision and resets, through engineering [27] or manual human intervention [56, 69, 70]. To make real-world training more broadly applicable (e.g., in the outdoors) we perform all state estimation on board, without any motion capture or external perception. While these prior works have demonstrated learning very conservative walking gaits on simple robots in the real world from scratch, we demonstrate learning of pacing and side-stepping, behaviors that are naturally unstable and require careful balancing on a more agile A1 robot. Thus, we found that it was crucial to use motion imitation and adopt a real-world fine-tuning approach rather than learning completely from scratch. Lastly, rather than manually resetting the robot or hand-designing a recovery controller specific to the robot, we used RL to automatically produce a reset controller, an approach that can be applied to automatically produce reset controllers for other quadrupeds as well.

Algorithm 1 `TRAIN`: RL Subroutine

Require: Critic to actor update ratio K
Require: $\mathcal{M}_i, Q_{\theta_i}, \pi_i, \mathcal{D}_i$
// Collect data
Compute goal $\mathbf{g}_t$ from $\mathcal{M}_i$
Collect trajectory τ with $\pi_i(\mathbf{a} \mid \mathbf{s}, \mathbf{g})$
Store τ in $\mathcal{D}_i$
// Perform updates
for $i = 1$ to n_{updates} **do**
    Update Q_{θ_i} via $\mathcal{L}^{\texttt{REDQ}}_{\texttt{critic}}$
    if $i \mod K == 0$ **then**
        Update π_i via $\mathcal{L}^{\texttt{REDQ}}_{\texttt{actor}}$
return $\mathcal{M}_i, Q_{\theta_i}, \pi_i, \mathcal{D}_i$

Algorithm 2 Real-World Fine-Tuning with Pre-training

Require: N_{refs} reference motions $\{\mathcal{M}_i\}_{i=1}^{N_{\text{refs}}}$
Initialize: $\{Q_{\theta i}, \pi_i, \mathcal{D}_i\}_{i=1}^{N_{\text{refs}}+1}$, skills $\mathcal{S}$
// Pre-training in simulation
for skill $i = 0, 2, \ldots, N_{\text{refs}}$ **do**
    repeat
        $S_i \leftarrow \texttt{TRAIN}(\mathcal{M}_i, Q_{\theta_i}, \pi_i, \mathcal{D}_i)$
    until convergence
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_i\}$
// Real-world fine-tuning
repeat
    Choose skill S_i
    if in new environment **then**
        Optionally clear $\mathcal{D}_i$
    $S_i \leftarrow \texttt{TRAIN}(\mathcal{M}_i, Q_{\theta_i}, \pi_i, \mathcal{D}_i)$
until forever

2.3 Fine-Tuning Locomotion in the Real World

In this section, we present our system for fine-tuning locomotion policies. It combines a stable and efficient RL algorithm with multi-task training, thereby allowing our robot to learn quickly with minimal human intervention. Sample-efficient learning is achieved by using a recently proposed off-policy RL algorithm, randomized ensembled double Q-learning (REDQ), which has demonstrated efficient learning in simulated environments [58]. To enable autonomous training in the real world without requiring human intervention, we stitch together episodes with a learned reset policy.

Overview Our framework, shown in Figure 2.2, involves learning a set of policies, one for each desired skill. Because RL algorithms are data intensive and untrained policies can be dangerous on a physical robot, we pre-train our policies in simulation, as is typical for legged locomotion controllers [28, 32, 71]. In this phase, we independently train a policy π_i for each of the skills, including a recovery policy. Once the policies are pre-trained in simulation, we perform fine-tuning in the real world by simply continuing the training process using the same RL algorithm. Because the dynamics may be significantly different in the real world, we reset the replay buffers $\mathcal{D}$ for each policy. After each episode, the learned recovery policy resets the robot in preparation for the next rollout, preventing time- and labor-intensive manual resets. For some skills, we use a multi-task framework, which leverages multiple skills to further facilitate autonomous learning. Algorithm 2 provides an overview of the complete training process.

Motion imitation Our policies are trained to perform different skills by imitating reference motion clips using the RL framework proposed by Peng et al. [57]. Given a reference motion $\mathcal{M}$ comprising a sequence of poses, a policy is trained to imitate the motion using a reward function that encourages tracking the target poses at each timestep (see Subsection 2.4.2). This general framework allows us to learn different skills simply by swapping out the reference motion. We also learn a recovery policy within this framework by training the robot to imitate a standing pose along with a few important modifications; see Subsection 2.4.3 for details.

Off-policy RL We leverage the off-policy REDQ algorithm [58], a simple extension to SAC [72] that allows for a larger ratio of gradient steps to time steps, for sample-efficient RL. REDQ utilizes an ensemble of Q-functions $Q_\theta = \{Q_{\theta^k}\}_{k=1}^{N_{\text{ensemble}}}$ that are all trained with respect to the same target value, which is computed by minimizing over a random subset of the ensemble. This avoids overestimation issues that can occur when using too many gradient steps. Our update procedure is summarized in Algorithm 1.

2.4 System Design

We use the A1 robot from Unitree as our robot platform and build our simulation using PyBullet [73]. For our motion imitation skills, we retarget a mocap recording of dog pacing from a public dataset [74] and an artist generated side-step motion for the A1 using inverse-kinematics (see [57]). The policies $\{\pi_i\}_{i=1}^{N_{\text{refs}}}$ and Q-functions $\{Q_{\theta_i}\}_{i=1}^{N_{\text{refs}}}$ are modeled using separate fully-connected neural networks. Updates are computed using the Adam optimizer [75] with a learning rate of 10^{-4} and a batch size of 256 transitions. All networks are constructed and trained using TensorFlow [76].

2.4.1 State and Action Spaces

The state $\mathbf{s}_t$ contains a history of 3 timesteps for each of the following features: root orientation (read from the IMU), joint angles, and previous actions. Similar to prior motion imitation approaches [54, 57], the policy receives not only proprioceptive input but a goal $\mathbf{g}_t$, which comprises the target poses (root position, root rotation, and joint angles) calculated from the reference motion for future timesteps. In our experiments, we use 4 future target poses, the latest of which is a target for approximately 1 second ahead of the current timestep. Actions $\mathbf{a}_t$ are PD position targets for each of the 12 joints and applied at a frequency of 33Hz. To ensure smoothness of the motions, we process the PD targets with a low-pass filter before supplying them to the robot.

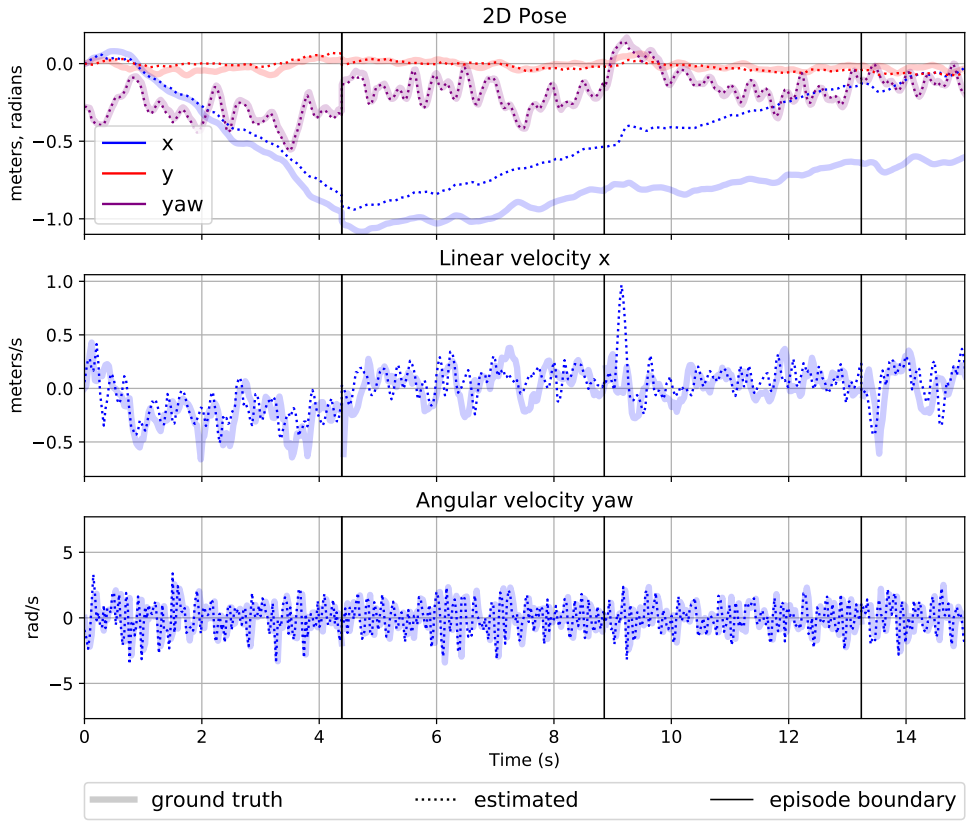


Figure 2.3: **(Analysis of on-board state estimation quality)** Real-world state estimation compared to motion capture for a robot walking indoors. Yaw and yaw velocity are very accurate, linear velocity is acceptable, and x- and y-position drift over the course of multiple episodes.

2.4.2 Reward Function

We adopt the reward function from [57], where the reward r_t at each timestep is calculated according to:

$$r_t = w^{\mathrm{p}} r_t^{\mathrm{p}} + w^{\mathrm{v}} r_t^{\mathrm{v}} + w^{\mathrm{e}} r_t^{\mathrm{e}} + w^{\mathrm{rp}} r_t^{\mathrm{rp}} + w^{\mathrm{rv}} r_t^{\mathrm{rv}} \tag{2.1}$$

$$w^{\mathrm{p}} = 0.5,\ w^{\mathrm{v}} = 0.05,\ w^{\mathrm{e}} = 0.2,\ w^{\mathrm{rp}} = 0.15,\ w^{\mathrm{rv}} = 0.1$$

The pose reward r_t^{p} encourages the robot to match its joint rotations with those of the reference motion. Below, $\hat{q}_t^j$ represents the local rotation of joint j from the reference motion at time t, and q_t^j represents the robot's joint,

$$r_t^{\mathrm{p}} = \exp\left[-5 \sum_j ||\hat{q}_t^j - q_t^j||^2\right]. \tag{2.2}$$

Figure 2.4: **Recovery controller training conditions** Examples from the initial state distribution used to train the reset policy in simulation. To get these states, we drop the robot from about a half meter above the ground with a random initial orientation of the robot's torso. Specifically, its roll, pitch and yaw are drawn from uniform distributions over the intervals $[-\frac{3\pi}{4}, \frac{3\pi}{4}]$, $[-\frac{\pi}{4}, \frac{\pi}{4}]$, and $[-\pi, \pi]$, respectively.

r_t^{v} and r_t^{e} assume a similar form but encourage matching the joint velocities and end-effector positions, respectively. Finally, the root pose reward r_t^{rp} and root velocity reward r_t^{rv} encourage the robot to track the reference root motion. See [57] for a detailed description of the reward function.

To estimate the linear root velocity during real-world training, we use a Kalman filter that takes acceleration and orientation readings from the IMU, then corrects them with the foot contact sensors. When a sensor is triggered, we take that foot as a point of 0 velocity, calculate the root velocity using the leg joint velocities, and correct the estimate from the IMU. We integrate the linear velocity for a rough approximation of the robot's position. Some example data is shown in Figure 2.3. We find that the angular velocity and orientation readings are very accurate, linear velocity is reasonable, and position drifts but is good enough within each episode for our reward calculations.

2.4.3 Reset Controller

Similar to [77], the reset policy is trained in simulation by generating a diverse set of initial states. At the start of each episode, we drop the robot from a random height and orientation (see Figure 2.4). The robot's objective then is to recover back to a default standing pose. [77] stitches together two policies for reset, first, a self-righting behavior which puts the robot in a stable sitting position, followed by a stand-up behavior which then puts the robot in a standing pose.

We find that we are able to train a single, streamlined reset policy by modifying the motion imitation objective. Rather than using a reference motion to prescribe exactly how the robot should stand up, we modify our standard imitation reward as follows. First, the policy is only rewarded for rolling right side up. If the robot is upright, we add the motion imitation reward, where the reference is a standing pose, to encourage the robot to stand. Note that although the objective for this reset policy is simple, the behaviors that it acquires are quite complex and agile. This policy is significantly more versatile than hand-designed recoveries used in prior work [27] or company-provided reset

motions that take more than 10 seconds — it can recover quickly from a fall by rolling and jumping upright, and if the robot is already upright, it quickly stabilizes it for the next trial. We encourage the reader to review videos of the agile reset controller here: https://sites.google.com/berkeley.edu/fine-tuning-locomotion. We found that the reset policy transfers successfully to all our test terrains, so we perform no fine-tuning.

2.5 Experiments

We aim to answer the following through our experiments:

(1) How does our finetuning-based method compare to prior approaches that utilize simulated training, including those that perform real-world adaptation?

(2) What effects do our design decisions have on the feasibility of real-world training?

(3) How much can autonomous, online fine-tuning improve robotic skills in a range of real-world settings?

2.5.1 Simulation Experiments

To compare our approach to prior methods, we evaluate on a simulated transfer scenario, where the policy is first trained in one simulated environment, and then "deployed" to another simulated setting, which is meant to be representative of the kind of domain shifts we would expect in real-world deployment. Performing this comparison in simulation allows us to abstract away other parts of the adaptation process, such as resets, since prior methods generally do not handle this, and allows us to provide highly controlled low-variance comparisons for each method.

Comparison to prior work To address (1), we compare our fine-tuning method to prior work, adapting a learned forward pacing gait to several test environments. To this end, we pre-train all methods with standard dynamics randomization (varying mass, inertia, motor strength, friction, latency) on flat ground until convergence, and then deploy them on various terrains for adaptation. The test terrains include a flat ground terrain that is similar to the environments seen during pre-training, as well as two terrains that differ substantially from the training settings: randomized heightfield simulating rugged terrain, and a low-friction surface simulating a slippery, icy terrain (with a friction coefficient far below that seen during training). Note that we intentionally select these test settings to be different from the training environment: while prior works generally carefully design the training environments to cover the range of settings seen at test-time, we intentionally want to evaluate the methods under *unexpected* conditions. For each type of environment, we use 10 instantiations of the environment with different

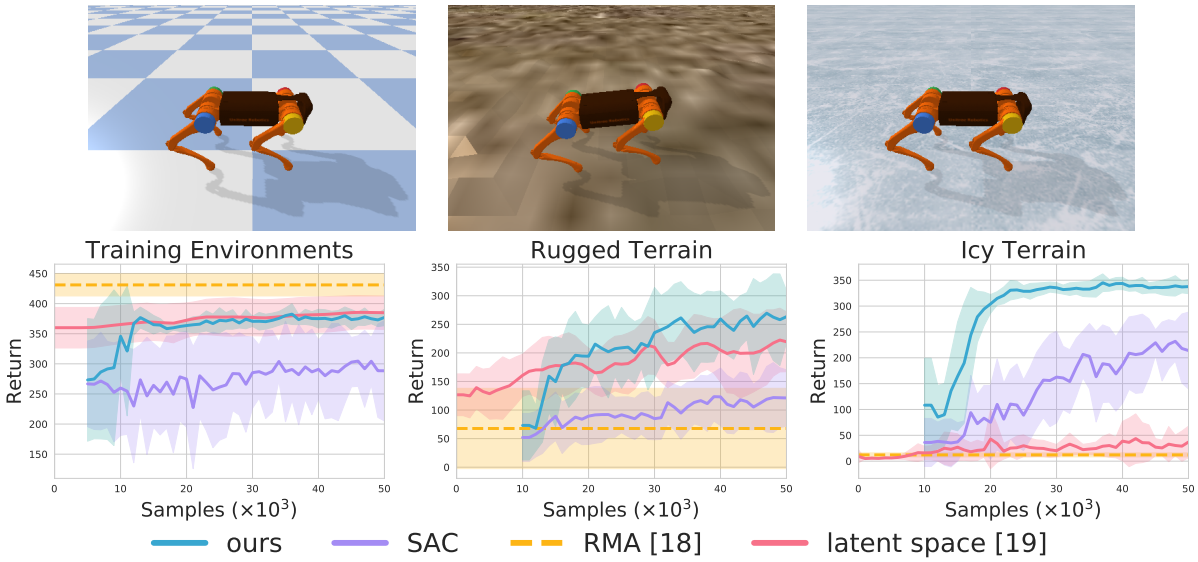


Figure 2.5: **(Simulation fine-tuning analysis)** We report each adapted policy's performance (mean and standard deviation across 10 trials) in the target domain (pictured in the top row) with respect to the samples used for adaptation. Prior methods that learn adaptation strategies during training (pink and yellow) excel in environments that are similar to those seen during training (left) but fail in environments that are sufficiently different (middle, right). In contrast, our fine-tuning method (blue) continues to improve under all circumstances. We also note that our use of REDQ (blue) improves over SAC (purple).

dynamics parameters, and test each method's ability to adapt to them. The dynamics of the environment is fixed for each adaptation trial, and the same 10 environments are used for all methods.

Peng et al. [57] uses dynamics randomization to learn a latent representation of behaviors that are effective for different settings of the dynamics parameters. During adaptation, this method searches in the latent space for a behavior that maximizes the agent's test-time performance using AWR [78]. Rapid Motor Adaptation (RMA) [34] also uses dynamics randomization to learn a latent representation of different strategies. But instead of searching in latent space during adaptation, RMA trains an additional 'adaptation module,' similarly to Yu et al. [29, 67], to predict the appropriate latent encoding given a recent history of observations and actions. Since RMA corresponds to a policy with memory and does not actually use data collected from a new domain to update the parameters of the model, it does not improve with additional data. We implement both methods using SAC as the policy optimizer at training-time, to match our method.

REDQ vs. SAC One of our design decisions is to use the recently-proposed REDQ algorithm for policy optimization. To evaluate the importance of this choice, we consider an

ablation of our method that uses vanilla SAC instead of REDQ for both pre-training and fine-tuning. For REDQ, we use 10 Q-functions, and randomly sample 2 when computing the target value. For both fine-tuning methods, we collect an initial buffer of samples before starting to fine-tune (hence these curves start at 5000 or 10000 samples).

We report the adaptation performance of each method in Figure 2.5. When tested on the training environments, we see that both RMA and the latent space method perform well. This indicates that these methods indeed excel in regimes where the environments seen at training-time resemble the ones that the method must adapt to at test-time. However, when these policies, which are trained on flat ground, are placed on uneven or extremely slippery terrain, they exhibit a significant drop in performance. Both methods suffer because they assume that a pre-trained encoder or latent space can generalize, which is too strong an assumption when the test environment differs sufficiently from the training environments. RMA especially suffers in this case because it relies entirely on the pre-trained encoder to adapt, and when this encoder fails to generalize, it does not have any other recourse. The latent space method does adapt, but it relies on the latent space already containing suitable strategies for the new environment and ends up with a suboptimal policy. In contrast, our finetuning approach is able to continuously improve and eventually succeed. These results show that pretrained models, even prior adaptive models, can fail if tested on environments that deviate too much from those seen in training. Our method initially fails also, but is able to recover good performance through fine-tuning.

2.5.2 Real-World Experiments

Our real-world experiments aim to evaluate how much autonomous online fine-tuning can improve a variety of robotic skills in a range of realistic settings. We evaluate our fine-tuning system in four real-world domains: an outdoor grassy lawn, a carpeted room, a doormat with crevices, and memory foam (see Figure 2.7), each of which presents unique challenges. The outdoor lawn presents a slippery surface, where the feet can either slip on the grass or get stuck in the dirt. In this domain, we finetune a pacing gait in which both legs on one side of the body swing forward in unison. For the indoors experiments, the robot is tasked with performing a side-stepping motion on the various surfaces mentioned above. The carpeted room, in contrast to the grass, is high-friction, causing the robot's soft rubber feet to deform in a manner inconsistent with simulation. The doormat presents a textured surface for the feet to get stuck. The 4cm-thick memory foam mattress is especially difficult, because the feet can sink into the mattress and the gaits need to change substantially to achieve ground clearance. For all experiments, we pretrain the policy on flat ground in simulation, run it for 5000 samples to initialize the buffer, and then finetune the policy in the real world.

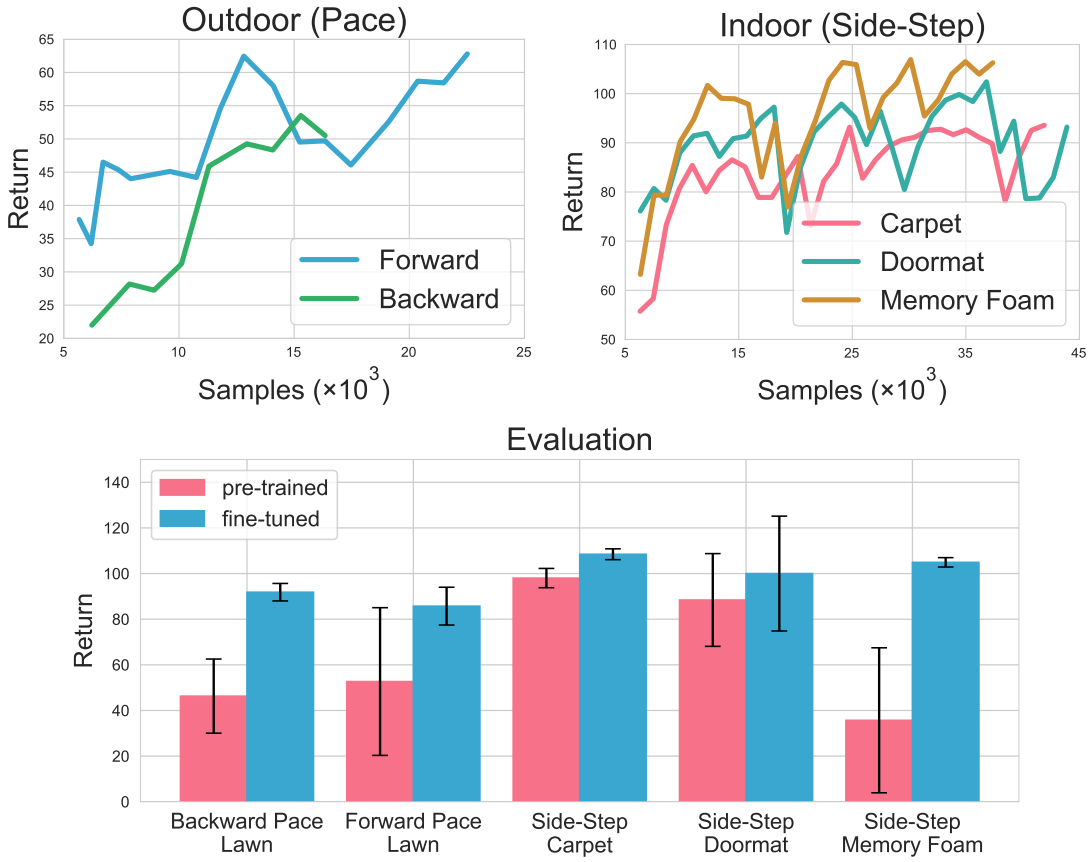


Figure 2.6: **(Real-world fine-tuning results)** Top: Learning curves for all the real-world fine-tuning experiments showing the average return of data collected by a stochastic policy during each iteration of training. Bottom: We evaluate each policy with a deterministic policy before and after fine-tuning and report mean and standard deviation over ten trials. In all domains, fine-tuning leads to improvement, and the improvement is particularly pronounced on the most difficult surfaces, such as the lawn and memory foam mattress. See the project website for videos of each of these policies throughout training.

We report the average return of the policies during training in Figure 2.6 with respect to the number of real-world samples collected. In all environments, our framework leads to substantial performance improvement with a modest amount of data. On the lawn, the pre-trained forward pacing policy makes very little forward progress, whereas the pre-trained backward pacing policy tends to trip and fall. After less than 2 hours in total of operation, the robot learns to consistently and stably pace forward and backward with very few failures. Indoors, the pre-trained sidestepping policy tends to twitch violently and fall to the floor before completing its motion. This is true across all the terrains: carpet, memory foam, and doormat with crevices. But on each terrain, in less than 2.5 hours of training, the robot learns to consistently execute the skill without stumbling.

This figure includes overhead such as swapping batteries and handling robot malfunctions. Figure 2.7 shows a comparison of the behaviors of the policy before and after fine-tuning. For all experiments, we include videos of the training process as well as evaluation of the performance before and after training on the project website.

Semi-autonomous training In our experiments, we find that our learned recovery controller is very effective in providing efficient, automatic resets in between trials. Over all experiments, the recovery policy was 100% successful. We compare our reset controller to the built-in rollover controller from Unitree. On hard surfaces, both controllers are effective while the built-in one is substantially slower than the learned policy. On the memory foam, the built-in controller performs less reliably. Videos of the reset policy in all environments and a comparison to the built-in controller are at . When the robot occasionally drifts outside the workspace area, it is re-positioned by a human supervisor. The need for this intervention, though, was greatly reduced by the use of the simultaneous learning of complementary skills.

2.6 Discussion and Limitations

We present a system that enables legged robots to fine-tune locomotion policies in real-world settings, such as grass, carpets, doormats and mattresses, via a combination of autonomous data collection and data-efficient model-free RL. Our system provides for automated recoveries from falls, reward calculation through state estimation using onboard sensors, and data-efficient fine-tuning of a variety of locomotion skills. The fine-tuning improves the performance substantially, reaching a high level of proficiency even when starting with a gait that frequently stumbles and falls. In this chapter, we focus on fine-tuning in each environment separately. It would be interesting to adapt our system into a lifelong learning process, where a robot never stops learning. When the robot encounters a new environment, the fine-tuning will quickly adapt its policy. When the robot stays in the same environment for an extended period of time, the fine-tuning will gradually perfect its skills. We plan to test such a lifelong learning system for legged robots in complex, diverse and ever-changing real-world environments.

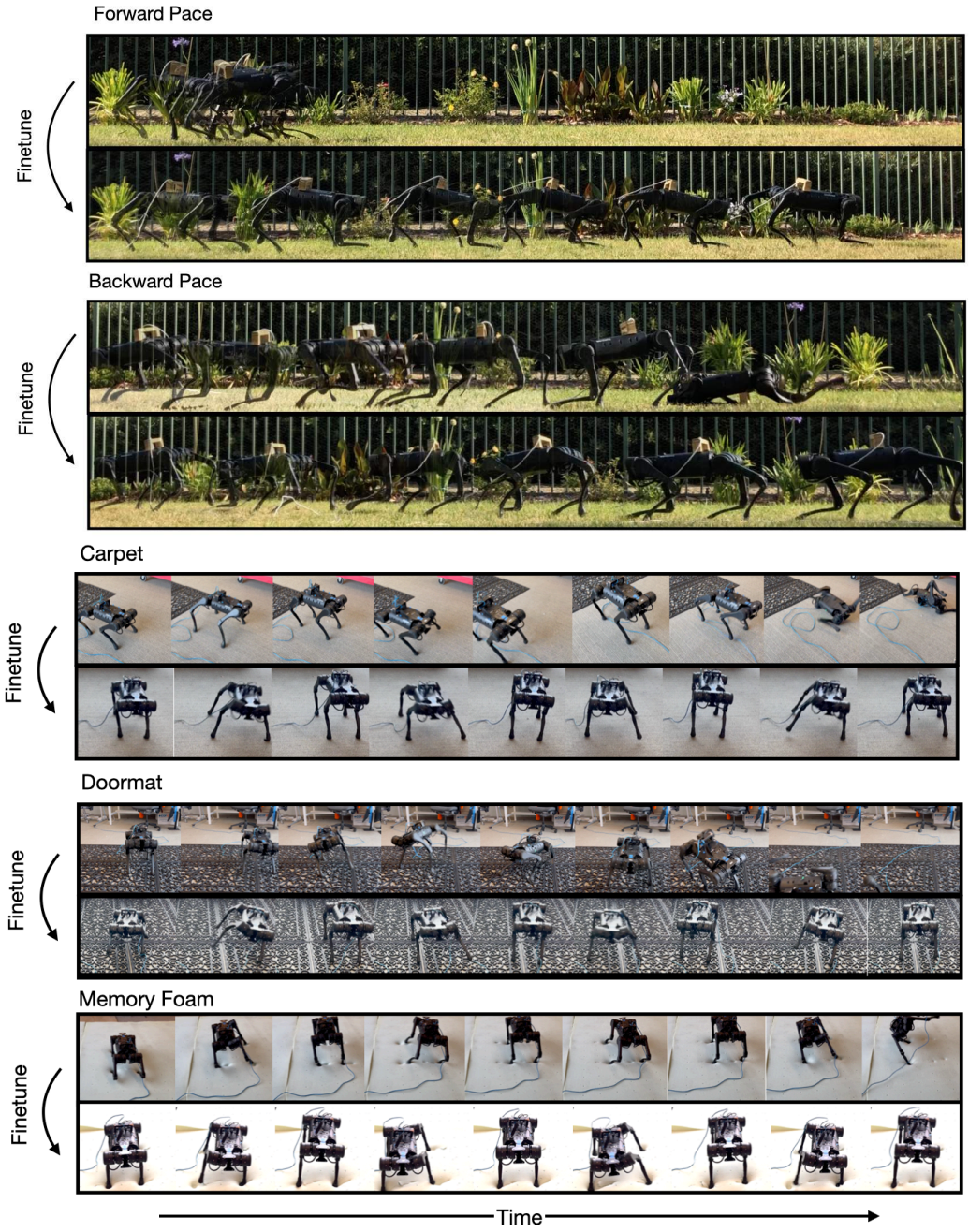


Figure 2.7: **(Example rollouts)** 1st and 2nd Rows: Example rollouts of the pacing policies before and after fine-tuning. The figure depicts a timelapse of a rollout of each policy, where the opacity of the robot indicates time progression (i.e., more opaque corresponds to more recent). Before fine-tuning, the forward policy struggles to make progress, while the pre-trained backwards policy makes fast progress but is quite unstable. After fine-tuning, both policies make forward progress without falling. 3rd to 5th Rows: Example rollouts of the side-step policies before and after fine-tuning. On all terrains, the pre-trained policies fail to complete the task without falling. After fine-tuning, the policies reliably execute the skill on all three domains.

Chapter 3

Feedback-Efficient Interactive RL

3.1 Motivation

Deep reinforcement learning (RL) has emerged as a powerful method whereby agents learn complex behaviors on their own through trial and error [22, 24, 79–82]. Scaling RL to many applications, however, is yet precluded by a number of challenges. One such challenge lies in providing a suitable reward function. For example, while it may be desirable to provide sparse rewards out of ease, they are often insufficient to train successful RL agents. Thus, to provide adequately dense signal, real-world problems may require extensive instrumentation, such as accelerometers to detect door opening [83], thermal cameras to detect pouring [84] or motion capture for object tracking [57, 85, 86].

Despite these costly measures, it may still be difficult to construct a suitable reward function due to reward exploitation. That is, RL algorithms often discover ways to achieve high returns by unexpected, unintended means. In general, there is nuance in how we might want agents to behave, such as obeying social norms, that are difficult to account for and communicate effectively through an engineered reward function [87–89]. A popular way to avoid reward engineering is through imitation learning, during which a learner distills information about its objectives or tries to directly follow an expert [90–93]. While imitation learning is a powerful tool, suitable demonstrations are often prohibitively expensive to obtain in practice [5, 94–96].

In contrast, humans often learn fairly autonomously, relying on occasional external feedback from a teacher. Part of what makes a teacher effective is their ability to interactively guide students according to their progress, providing corrective or increasingly advanced instructions as needed. Such an interactive learning process is also alluring for artificial agents since the agent's behavior can naturally be tailored to one's preference (avoiding reward exploitation) without requiring extensive engineering. This approach is only feasible if the feedback is both practical for a human to provide and sufficiently high-bandwidth.

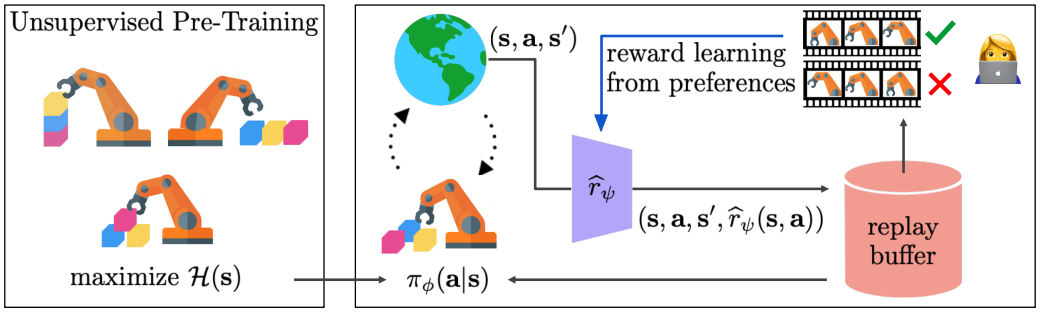


Figure 3.1: **(Overview of PEBBLE)** First, the agent engages in unsupervised pre-training during which it is encouraged to visit a diverse set of states so its queries can provide more meaningful signal than on randomly collected experience (left). Then, a teacher provides preferences between two clips of behavior, and we learn a reward model based on them. The agent is updated to maximize the expected return under the model. We also relabel all its past experiences with this model to maximize their utilization to update the policy (right).

As such, human-in-the-loop (HiL) RL [35, 97, 98] has not yet been widely adopted.

We aim to substantially reduce the amount of human effort required for HiL learning. To this end, we present PEBBLE: unsupervised **P**r**E**-training and preference-**B**ased learning via rela**B**e**L**ing **E**xperience, a feedback-efficient RL algorithm by which learning is largely autonomous and supplemented by a practical number of binary labels (i.e. preferences) provided by a supervisor. Our method relies on two main, synergistic ingredients: *unsupervised pre-training* and *off-policy learning* (see Figure 3.1). For generality, we do not assume the agent is privy to rewards from its environment. Instead, we first allow the agent to explore using only intrinsic motivation [99, 100] to diversify its experience and produce coherent behaviors. Collecting a breadth of experiences enables the teacher to provide more meaningful feedback, as compared to feedback on data collected in an indeliberate manner. The supervisor then steps in to teach the agent by expressing their preferences between pairs of clips of the agent's behavior [35]. The agent distills this information into a reward model and uses RL to optimize this inferred reward function.

Leveraging unsupervised pre-training increases the efficiency of the teacher's initial feedback; however, RL requires a large enough number of samples such that supervising the learning process is still quite expensive for humans. It is thus especially critical to enable off-policy algorithms that can reuse data to maximize the agent's, and thereby human's, efficiency. However, on-policy methods have typically been used thus far for HiL RL because of their ability to mitigate the effects of non-stationarity in reward caused by online learning. We show that by simply relabeling all of the agent's past experience every time the reward model is updated, we can make use and reuse of *all* the agent's collected experience to improve sample and feedback efficiency by a large margin. Source code and

videos are available at https://sites.google.com/view/icml21pebble.

We summarize the main contributions of PEBBLE:

- For the first time, we show that *unsupervised pre-training* and *off-policy learning* can significantly improve the sample- and feedback-efficiency of HiL RL.
- PEBBLE outperforms prior preference-based RL baselines on complex locomotion and robotic manipulation tasks from DeepMind Control Suite (DMControl; Tassa et al. 101, 102) and Meta-world [103].
- We demonstrate that PEBBLE can learn behaviors for which a typical reward function is difficult to engineer very efficiently.
- We also show that PEBBLE can avoid reward exploitation, leading to more desirable behaviors compared to an agent trained with respect to an engineered reward function.

3.2 Related Work

Learning from human feedback Several works have successfully utilized feedback from real humans to train agents where it is assumed that the feedback is available at *all* times [98, 104, 105]. Due to this high feedback frequency, these approaches are difficult to scale to more complex learning problems that require substantial agent experience.

Better suited to learning in complex domains is to learn a reward model so the agent can learn without a supervisor's perpetual presence. One simple yet effective direction in reward learning is to train a classifier that recognizes task success and use it as basis for a reward function [25, 106–108]. Positive examples may be designated or reinforced through human feedback [109–111]. Another promising direction has focused on simply training a reward model via regression using unbounded real-valued feedback [97, 112], but this has been challenging to scale because it is difficult for humans to reliably provide a particular utility value for certain behaviors of the RL agent.

Much easier for humans is to make *relative* judgments, i.e., comparing behaviors as better or worse. Preference-based learning is thus an attractive alternative because the supervision is easy to provide yet information-rich [104, 113–122]. Christiano et al. [35] scaled preference-based learning to utilize modern deep learning techniques—they learn a reward function, modeled with deep neural networks, that is consistent with the observed preferences and use it to optimize an agent using RL. They choose on-policy RL methods [123, 124] since they are more robust to the non-stationarity in rewards caused by online learning. Although they demonstrate that preference-based learning provides a fairly efficient (requiring feedback on less than 1% of the agent's experience) means of distilling information from feedback, they rely on notoriously sample-inefficient on-policy

RL, so a large burden can yet be placed on the human. Subsequent works have aimed to improve the efficiency of this method by introducing additional forms of feedback such as demonstrations [125] or non-binary rankings [126]. Our proposed approach similarly focuses on developing a more sample- and feedback-efficient preference-based RL algorithm *without* adding any additional forms of supervision. Instead, we enable off-policy learning as well as utilize *unsupervised* pre-training to substantially improve efficiency.

Unsupervised pre-training for RL Unsupervised pre-training has been studied for extracting strong behavioral priors that can be utilized to solve downstream tasks efficiently in the context of RL [127–131]. Specifically, agents are encouraged to expand the boundary of seen states by maximizing various intrinsic rewards, such as prediction errors [132–134], count-based state novelty [135–137], mutual information [130] and state entropy [138–140]. Such unsupervised pre-training methods allow learning diverse behaviors without extrinsic rewards, effectively facilitating accelerated learning of downstream tasks. In this chapter, we show that unsupervised pre-training enables a teacher to provide more meaningful signal by showing them a diverse set of behaviors.

3.3 Preliminaries

Reinforcement learning We consider a standard RL framework where an agent interacts with an environment in discrete time. Formally, at each timestep t, the agent receives a state $\mathbf{s}_t$ from the environment and chooses an action $\mathbf{a}_t$ based on its policy π. The environment returns a reward r_t and the agent transitions to the next state $\mathbf{s}_{t+1}$. The return $\mathcal{R}_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the discounted sum of rewards from timestep t with discount factor $\gamma \in [0, 1)$. RL then maximizes the expected return from each state $\mathbf{s}_t$.

Soft Actor-Critic SAC [72] is an off-policy actor-critic method based on the maximum entropy RL framework [141], which encourages exploration and greater robustness to noise by maximizing a weighted objective of the reward and the policy entropy. To update the parameters, SAC alternates between a soft policy evaluation and a soft policy improvement. At the soft policy evaluation step, a soft Q-function, which is modeled as a neural network with parameters θ, is updated by minimizing the following soft Bellman residual:

$$\begin{aligned} &\mathcal{L}_{\texttt{critic}}^{\texttt{SAC}} = \mathbb{E}_{\tau_t \sim \mathcal{B}}\Big[\big(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r_t - \gamma \bar{V}(\mathbf{s}_{t+1})\big)^2 \Big], \\ &\text{with} \quad \bar{V}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\big[Q_{\bar{\theta}}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) \big], \end{aligned} \tag{3.1}$$

where $\tau_t = (\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ is a transition, $\mathcal{B}$ is a replay buffer, $\bar{\theta}$ are the delayed parameters, and α is a temperature parameter. At the soft policy improvement step, the policy π_ϕ is

updated by minimizing the following objective:

$$\mathcal{L}_{\texttt{act}}^{\texttt{SAC}} = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{B}, \mathbf{a}_t \sim \pi_\phi} \Big[\alpha \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \Big]. \tag{3.2}$$

SAC enjoys good sample efficiency relative to on-policy methods by reusing its past experiences. However, for the same reason, SAC is not robust to a non-stationary reward function.

Reward learning from preferences We follow the basic framework for learning a reward function $\widehat{r}_\psi$ from preferences in which the function is trained to be consistent with observed human feedback [35, 115].

In this framework, a segment σ is a sequence of observations and actions $\{\mathbf{s}_k, \mathbf{a}_k, ..., \mathbf{s}_{k+H}, \mathbf{a}_{k+H}\}$. We elicit preferences y for segments σ^0 and σ^1, where y is a distribution indicating which segment a human prefers, i.e., $y \in \{(0,1), (1,0), (0.5, 0.5)\}$.

The judgment is recorded in a dataset $\mathcal{D}$ as a triple (σ^0, σ^1, y). By following the Bradley-Terry model [142], we model a preference predictor using the reward function $\widehat{r}_\psi$ as follows:

$$P_\psi[\sigma^1 \succ \sigma^0] = \frac{\exp \sum_t \widehat{r}_\psi(\mathbf{s}_t^1, \mathbf{a}_t^1)}{\sum_{i \in \{0,1\}} \exp \sum_t \widehat{r}_\psi(\mathbf{s}_t^i, \mathbf{a}_t^i)}, \tag{3.3}$$

where $\sigma^i \succ \sigma^j$ denotes the event that segment i is preferable to segment j.

Intuitively, this can be interpreted as assuming the probability of preferring a segment depends exponentially on the sum over the segment of an underlying reward function. While $\widehat{r}_\psi$ is not a binary classifier, learning $\widehat{r}_\psi$ amounts to binary classification with labels y provided by a supervisor. Concretely, the reward function, modeled as a neural network with parameters ψ, is updated by minimizing the following loss:

$$\begin{aligned} \mathcal{L}^{\texttt{Reward}} = - \mathop{\mathbb{E}}_{(\sigma^0, \sigma^1, y) \sim \mathcal{D}} \Big[& y(0) \log P_\psi[\sigma^0 \succ \sigma^1] \\ & + y(1) \log P_\psi[\sigma^1 \succ \sigma^0] \Big]. \end{aligned} \tag{3.4}$$

3.4 PEBBLE

In this section, we present PEBBLE: unsupervised **P**r**E**-training and preference-**B**ased learning via rela**B**e**L**ing **E**xperience, an off-policy actor-critic algorithm for HiL RL. Formally, we consider a policy π_ϕ, Q-function Q_θ and reward function $\widehat{r}_\psi$, which are updated by the following processes (see Algorithm 4 for the full procedure):

- *Step 0 (unsupervised pre-training)*: We pre-train the policy π_ϕ only using intrinsic motivation to explore and collect diverse experiences (see Section 3.4.1).

Algorithm 3 `EXPLORE`: Unsupervised exploration

Initialize parameters of Q_θ and π_ϕ and a replay buffer $\mathcal{B} \leftarrow \emptyset$
for each iteration **do**
  for each timestep t **do**
    Collect $\mathbf{s}_{t+1}$ by taking $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)$
    Compute intrinsic reward $r_t^{\texttt{int}} \leftarrow r^{\texttt{int}}(\mathbf{s}_t)$ as in (3.5)
    Store transitions $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t^{\texttt{int}})\}$
  for each gradient step **do**
    Sample minibatch $\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1}, r_j^{\texttt{int}})\}_{j=1}^{B} \sim \mathcal{B}$
    Optimize $\mathcal{L}^{\texttt{SAC}}_{\texttt{critic}}$ in (3.1) and $\mathcal{L}^{\texttt{SAC}}_{\texttt{act}}$ in (3.2) w.r.t. θ and ϕ
return $\mathcal{B}, \pi_\phi$

- *Step 1 (reward learning)*: We learn a reward function $\widehat{r}_\psi$ that can lead to the desired behavior by getting feedback from a teacher (see Section 3.4.2).
- *Step 2 (agent learning)*: We update the policy π_ϕ and Q-function Q_θ using an off-policy RL algorithm with relabeling to mitigate the effects of a non-stationary reward function $\widehat{r}_\psi$ (see Section 3.4.3).
- Repeat *Step 1* and *Step 2*.

3.4.1 Accelerating Learning via Unsupervised Pre-training

In our setting, we assume the agent is given feedback in the form of preferences between segments. In the beginning of training, though, a naive agent executes a random policy, which does not provide good state coverage nor coherent behaviors. The agent's queries are thus quite limited and likely difficult for human teachers to judge. As a result, it requires many samples (and thus queries) for these methods to show initial progress. Recent work has addressed this issue by means of providing demonstrations; however, this is not ideal since these are notoriously hard to procure [125]. Instead, our insight is to produce informative queries at the start of training by utilizing unsupervised pretraining to collect diverse samples solely through intrinsic motivation [99, 100].

Specifically, we encourage our agent to visit a wider range of states by using the state entropy $\mathcal{H}(\mathbf{s}) = -\mathbb{E}_{\mathbf{s}\sim p(\mathbf{s})}[\log p(\mathbf{s})]$ as an intrinsic reward [138–140, 143]. By updating the agent to maximize the sum of expected intrinsic rewards, it can efficiently explore an environment and learn how to generate diverse behaviors. However, this intrinsic reward is intractable to compute in most settings. To handle this issue, we employ the simplified version of particle-based entropy estimator [144, 145] (see the supplementary material for more details):

$$\widehat{\mathcal{H}}(\mathbf{s}) \propto \sum_i \log(||\mathbf{s}_i - \mathbf{s}_i^k||),$$

where $\widehat{\mathcal{H}}$ denotes the particle-based entropy estimator and $\mathbf{s}_i^k$ is the k-th nearest neighbor (k-NN) of $\mathbf{s}_i$. This implies that maximizing the distance between a state and its nearest neighbor increases the overall state entropy. Inspired by this, we define the intrinsic reward of the current state $\mathbf{s}_t$ as the distance between $\mathbf{s}_t$ and its k-th nearest neighbor by following the idea of Liu and Abbeel [140] that treats each transition as a particle:

$$r^{\texttt{int}}(\mathbf{s}_t) = \log(||\mathbf{s}_t - \mathbf{s}_t^k||). \tag{3.5}$$

In our experiments, we compute k-NN distances between a sample and all samples in the replay buffer and normalize the intrinsic reward by dividing it by a running estimate of the standard deviation. The full procedure of unsupervised pre-training is summarized in Algorithm 3.

3.4.2 Selecting Informative Queries

As previously mentioned, we learn our reward function by modeling the probability that a teacher prefers one sampled segment over another as proportional to the exponentiated sum of rewards over the segment (see Section 3.3). Ideally, one should solicit preferences so as to maximize *expected value of information* (EVOI; Savage 146): the improvement of an agent caused by optimizing with respect to the resulting reward model [114, 147]. Computing the EVOI is intractable since it involves taking an expectation over all possible trajectories induced by the updated policy. To handle this issue, several approximations have been explored by prior works to sample queries that are likely to change the reward model [35, 125, 148]. In this chapter, we consider the sampling schemes employed by Christiano et al. [35]: (1) uniform sampling and (2) ensemble-based sampling, which selects pairs of segments with high variance across ensemble reward models. We explore an additional third method, entropy-based sampling, which seeks to disambiguate pairs of segments nearest the decision boundary. That is, we sample a large batch of segment pairs and select pairs that maximize $\mathcal{H}(P_\psi)$. We evaluate the effects of these sampling methods in Section 3.5.

3.4.3 Using Off-policy RL with Non-Stationary Reward

Once we learn a reward function $\widehat{r}_\psi$, we can update the policy π_ϕ and Q-function Q_θ using any RL algorithm. A caveat is that the reward function $\widehat{r}_\psi$ may be non-stationary because we update it during training. Christiano et al. [35] used on-policy RL algorithms, TRPO [123] and A2C [124], to address this issue. However, their poor sample-efficiency leads to poor feedback-efficiency of the overall HiL method.

In this chapter, we use an off-policy RL algorithm, which provides for sample-efficient learning by reusing past experiences that are stored in the replay buffer. However, the learning process of off-policy RL algorithms can be unstable because previous experiences

Algorithm 4 PEBBLE

Require: Frequency of teacher feedback K
Require: Number of queries M per feedback session
Initialize parameters of Q_θ and $\widehat{r}_\psi$
Initialize a dataset of preferences $\mathcal{D} \leftarrow \emptyset$
// Exploration phase
$\mathcal{B}, \pi_\phi \leftarrow$ `EXPLORE`() ▷ See Algorithm 3
// Policy learning
for each iteration **do**
  // Reward learning
  if iteration mod $K = 0$ **then**
    for $m = 1$ to M **do**
      $(\sigma^0, \sigma^1) \sim$ `SAMPLE()` ▷ See Section 3.4.2
      Query instructor for y
      Store preference: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\sigma^0, \sigma^1, y)\}$
    for each gradient step **do**
      Sample minibatch $\{(\sigma^0, \sigma^1, y)_j\}_{j=1}^{D} \sim \mathcal{D}$
      Optimize $\mathcal{L}^{\texttt{Reward}}$ in (3.4) with respect to ψ
    Relabel entire replay buffer $\mathcal{B}$ using $\widehat{r}_\psi$
  for each timestep t **do**
    Collect $\mathbf{s}_{t+1}$ by taking $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)$
    Store transition: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \widehat{r}_\psi(\mathbf{s}_t))\}$
  for each gradient step **do**
    Sample minibatch $\{\tau_j\}_{j=1}^{B} \sim \mathcal{B}$
    Optimize $\mathcal{L}^{\texttt{SAC}}_{\texttt{critic}}$ in (3.1) and $\mathcal{L}^{\texttt{SAC}}_{\texttt{act}}$ in (3.2) w.r.t. θ and ϕ

in the replay buffer are labeled with previous learned rewards. To handle this issue, we relabel all of the agent's past experience every time we update the reward model. We find that this simple technique stabilizes the learning process and provides large gains in performance (see Figure 3.5a for supporting results). The full procedure of PEBBLE is summarized in Algorithm 4.

3.5 Experiments

We design our experiments to investigate the following:

1. How does PEBBLE compare to existing methods in terms of sample and feedback efficiency?
2. What is the contribution of each component in PEBBLE?
3. Can PEBBLE learn novel behaviors for which a typical reward function is difficult to engineer?
4. Can PEBBLE mitigate the effects of reward exploitation?

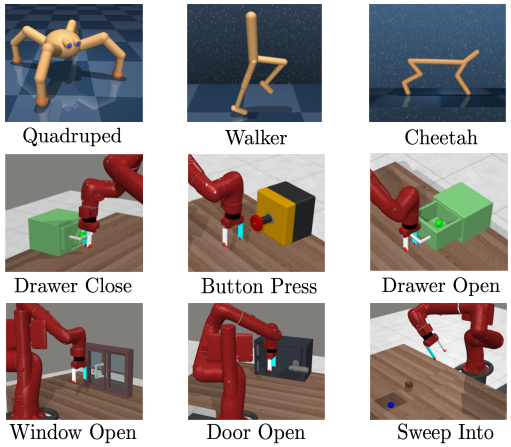


Figure 3.2: **(PEBBLE evaluation environments)** Examples from the environments we test on. We consider learning a variety of complex locomotion and manipulation skills through interacting with a scripted or human trainer.

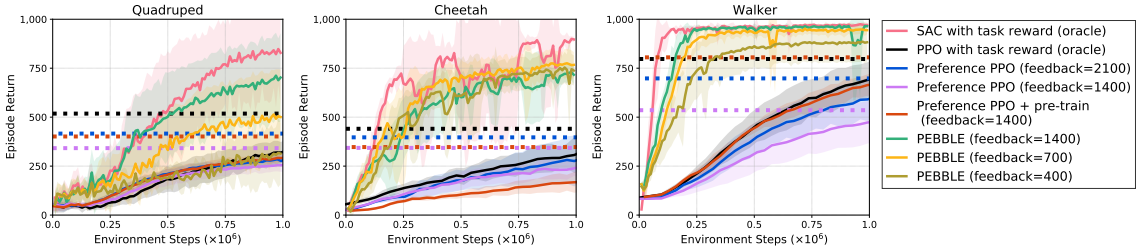


Figure 3.3: **(PEBBLE locomotion results)** Learning curves on locomotion tasks as measured on the ground truth reward. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and Preference PPO is indicated by dotted lines of the corresponding color.

3.5.1 Setups

We evaluate PEBBLE on several continuous control tasks involving locomotion and robotic manipulation from DeepMind Control Suite (DMControl; Tassa et al. 101, 102) and Meta-world [103]. In order to verify the efficacy of our method, we first focus on having an agent solve a range of tasks without being able to directly observe the ground truth reward function. Instead, similar to Christiano et al. [35] and Ibarz et al. [125], the agent learns to perform a task only by getting feedback from a scripted teacher that provides preferences between trajectory segments according to the true, underlying task reward. Because this scripted teacher's preferences are immediately generated by a ground truth

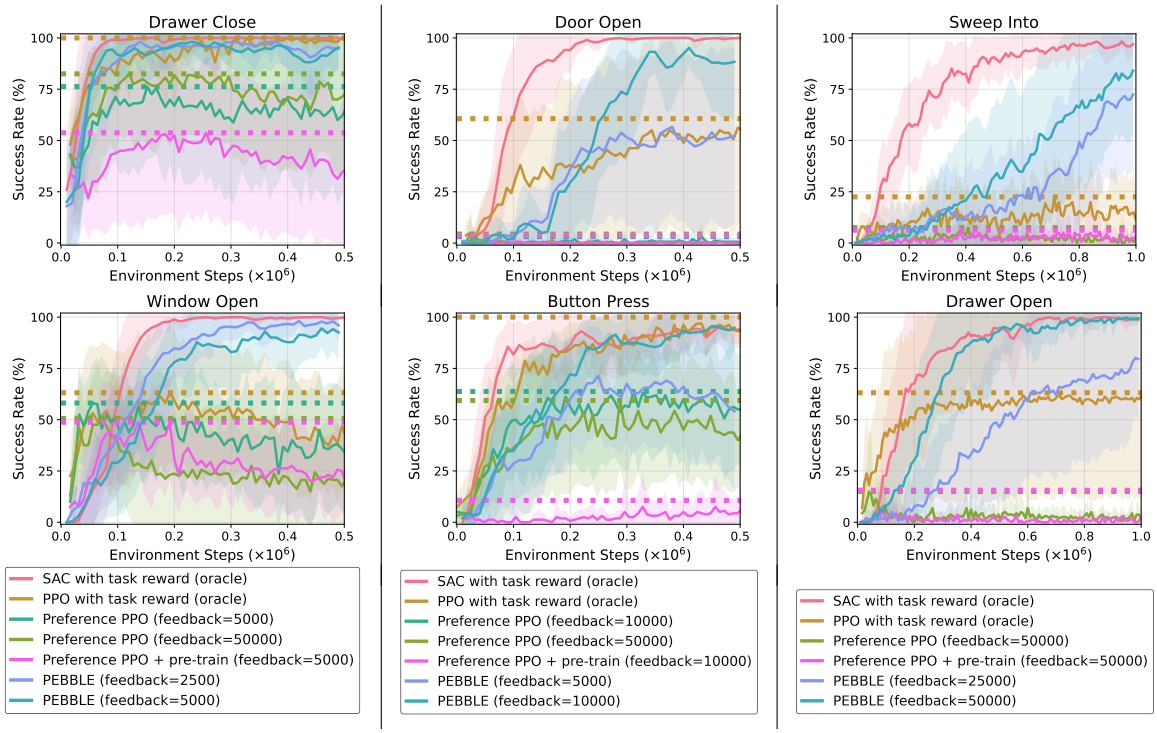
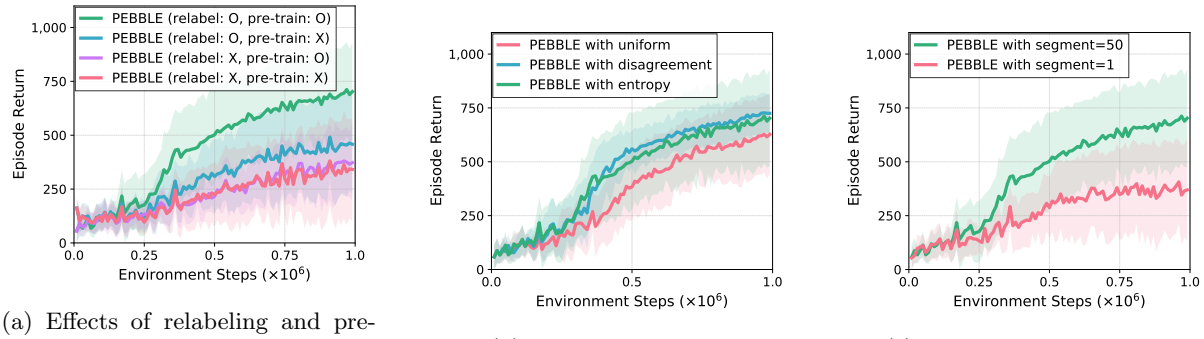


Figure 3.4: **(PEBBLE manipulation results)** Learning curves on robotic manipulation tasks as measured on the success rate. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs. Asymptotic performance of PPO and Preference PPO is indicated by dotted lines of the corresponding color.

reward, we are able to evaluate the agent quantitatively by measuring the true average return and do more rapid experiments. For all experiments, we report the mean and standard deviation across ten runs.

We also run experiments with actual human trainers (the authors) to show the benefits of human-in-the-loop RL. First, we show that human trainers can teach novel behaviors (e.g., waving a leg), which are not defined in original benchmarks. Second, we show that agents trained with the hand-engineered rewards from benchmarks can perform the task in an undesirable way (i.e., the agent exploits a misspecified reward function), while agents trained using human feedback can perform the same task in the desired way. For all experiments, each trajectory segment is presented to the human as a 1 second video clip, and a maximum of one hour of human time is required.

For evaluation, we compare to Christiano et al. [35], which is the current state-of-the-art approach using the same type of feedback. The primary differences in our method are (1) the introduction of unsupervised pre-training, (2) the accommodation of *off-policy*

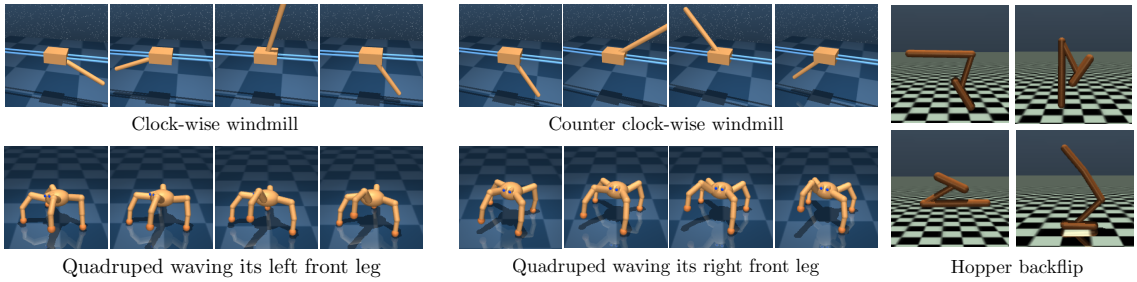


(a) Effects of relabeling and pre-training

(b) Sampling schemes

(c) Length of the segment

Figure 3.5: **(Ablation study on Quadruped-walk)** (a) Contribution of each technique in PEBBLE, i.e., relabeling the replay buffer (relabel) and unsupervised pre-training (pre-train). (b) Effects of sampling schemes to select queries. (c) PEBBLE with varying the length of the segment. The results show the mean and standard deviation averaged over ten runs.



Clock-wise windmill

Counter clock-wise windmill

Quadruped waving its left front leg

Quadruped waving its right front leg

Hopper backflip

Figure 3.6: **(Results using real human feedback)** Novel behaviors trained using feedback from human trainers. The corresponding videos and examples of selected queries are available in the supplementary material.

(a) Agent trained with human preference

(b) Agent trained with hand-engineered reward

Figure 3.7: **(Comparing human to simulated feedback)** Five frames from agents trained with (a) human preference and (b) hand-engineered reward from DMControl benchmark.

RL, and (3) entropy-based sampling. We re-implemented Christiano et al. [35] using the state-of-the-art on-policy RL algorithm: PPO [149]. We use the same reward learning framework and ensemble disagreement-based sampling as they proposed. We refer to this

baseline as Preference PPO.

As an upper bound, since we evaluate against the task reward function, we also compare to SAC [72] and PPO using the same ground truth reward. For our method, we pre-train an agent for 10K timesteps and include these pre-training steps in all learning curves. We do not alter any hyperparameters of the original SAC algorithm and use an ensemble of three reward models. Unless stated otherwise, we use entropy-based sampling. More experimental details including model architectures, sampling schemes, and reward learning are in the supplementary material.

3.5.2 Benchmark Tasks with Unobserved Rewards

Locomotion tasks from DMControl Figure 3.3 shows the learning curves of PEBBLE with 1400, 700 or 400 pieces of feedback[1] and that of Preference PPO with 2100 or 1400 pieces of feedback on three complex environments: Cheetah-run, Walker-walk and Quadruped-walk. Note that we explicitly give preference PPO an advantage by providing it with more feedback. We find that given a budget of 1400 queries, PEBBLE (green) reaches the same performance as SAC (pink) while Preference PPO (purple) is unable to match PPO (black). That PEBBLE requires less feedback than Preference PPO to match its respective oracle performance corroborates that PEBBLE is indeed more feedback-efficient. These results demonstrate that PEBBLE can enable the agent to solve the tasks without directly observing the ground truth reward function.

For further analysis, we incorporated our pre-training with Preference PPO (red) and find that it improves performance for Quadruped and Walker. We emphasize that our insight of using pre-training is able to improve both methods in terms of feedback-efficiency and asymptotic performance, but PEBBLE is uniquely positioned to benefit as it is able to utilize unsupervised experience for policy learning.

Robotic manipulation tasks from Meta-world One application area in which HiL methods could have significant real-world impact is robotic manipulation, since learning often requires extensive engineering in the real world [57, 83–86, 150]. However, the common approach is to perform goal-conditioned learning with classifiers [110], which can only capture limited information about what goal states *are*, and not about *how* they can be achieved. To study how we can utilize preference-based learning to perform more complex skills, we also consider six tasks covering a range of fundamental robotic manipulation skills from Meta-world (see Figure 3.2). As shown in Figure 3.4, PEBBLE matches the performance of SAC using the ground truth reward and outperforms Preference PPO, given comparable (and more) feedback, on every task. By demonstrating the applicability of PEBBLE to learning a variety of robotic manipulation tasks, we believe that we

[1] One piece of feedback corresponds to one preference query.

are taking an important step towards anyone (non-experts included) being able to teach robots in real-world settings.

3.5.3 Ablation Study

Contribution of each technique In order to evaluate the individual effects of each technique in PEBBLE, we incrementally apply unsupervised pre-training and relabeling. Figure 3.5a shows the learning curves of PEBBLE with 1400 queries on Quadruped-walk. First, we remark that relabeling significantly improves performance because it enables the agent to be robust to changes in its reward model. By additionally utilizing unsupervised pre-training, both sample-efficiency and asymptotic performance of PEBBLE are further improved because showing diverse behaviors to a teacher can induce a better-shaped reward. This shows that PEBBLE's key ingredients are fruitfully wed, and their unique combination is crucial to our method's success.

Effects of sampling schemes We also analyze the effects of different sampling schemes to select queries. Figure 3.5b shows the learning curves of PEBBLE with three different sampling schemes: uniform sampling, disagreement sampling and entropy sampling on Quadruped-walk. For this complex domain, we find that the uncertainty-based sampling schemes (using ensemble disagreement or entropy) are superior to the naive uniform sampling scheme. However, we note that they did not lead to extra gains on relatively simple environments, like Walker and Cheetah, similar to observations from Ibarz et al. [125] (see the supplementary material for more results).

Comparison with step-wise feedback We also measure the performance of PEBBLE by varying the length of segments. Figure 3.5c shows that feedback from longer segments (green curve) provide more meaningful signal than step-wise feedback (red curve). We believe that this is because longer segments can provide more context in reward learning.

3.5.4 Human Experiments

Novel behaviors We show that agents can perform various novel behaviors based on human feedback using PEBBLE in Figure 3.6. Specifically, we demonstrate (a) the Cart agent swinging a pole (using 50 queries), (b) the Quadruped agent waving a front leg (using 200 queries), and (c) the Hopper performing a backflip (using 50 queries). We note that the human is indeed able to guide the agent in a controlled way, as evidenced by training the same agent to perform several variations of the same task (e.g., waving different legs or spinning in opposite directions). The videos of all behaviors and examples of selected queries are available in the supplementary material.

Reward exploitation One concern in utilizing hand-engineered rewards is that an agent can exploit unexpected sources of reward, leading to unintended behaviors. Indeed, we find that the Walker agent learns to walk using only one leg even though it achieves the maximum scores as shown in Figure 3.7b. However, using 200 human queries, we were able to train the Walker to walk in a more natural, human-like manner (using both legs) as shown in Figure 3.7a. This result clearly shows the advantage of HiL RL to avoid reward exploitation.

3.6 Discussion

In this chapter, we present PEBBLE, a feedback-efficient algorithm for HiL RL. By leveraging unsupervised pre-training and off-policy learning, we show that sample- and feedback-efficiency of HiL RL can be significantly improved and this framework can be applied to tasks of higher complexity than previously considered by previous methods, including a variety of locomotion and robotic manipulation skills. Additionally, we demonstrate that PEBBLE can learn novel behaviors and avoid reward exploitation, leading to more desirable behaviors compared to an agent trained with respect to an engineered reward function. We believe by making preference-based learning more tractable, PEBBLE may facilitate broadening the impact of RL beyond settings in which experts can carefully craft reward functions to those in which laypeople can likewise utilize the advances of robot learning in the real world.

Part II

Learning in Real Time

Chapter 4

Learning to Walk in 20 Minutes With Model-Free RL

Figure 4.1: **(Overview of demonstration)** We demonstrate that deep reinforcement learning can be used to efficiently train a quadruped robot directly on various real world terrains, e.g., flat ground (blue); soft, irregular mulch (green); grass (red); and a hiking trail (yellow), acquiring effective gaits within 20 minutes of training.

4.1 Motivation

Agile, robust, and capable robotic skills require careful controller design and validation to work reliably in the real world. Reinforcement learning offers a promising alternative, acquiring effective control strategies directly through interaction with the real system, potentially right in the environment in which the robot will be situated. Although large-scale robotic RL experiments in the real world have been described in a number of prior works [151–155], many others have sought to sidestep the need for real-world training

over concerns about sample efficiency, often going to great lengths to design sophisticated simulation systems [12, 14, 65, 156, 157] and transfer learning methods [29, 31, 34, 57, 61, 67, 158–161]. For example, learning to solve a Rubik's cube with a robotic hand required 13 thousand years' worth of experience [13], which amounted to several months of wall-clock time using distributed training in simulation. In robotic locomotion, Rudin et al. [33] also utilize distributed training to collect 80 hours' worth of simulated experience to train an ANYmal robot to walk in 20 minutes. Using the A1 quadrupedal robot as we do, Kumar et al. [34] use 1.2 billion samples to train a robust controller, corresponding to roughly 4.5 months' worth of cumulative experience—without accounting for the often significant physical overhead required to train in the real world—that can be acquired in 1 day using an appropriate simulator. In this chapter, we focus specifically on the task of robotic quadrupedal locomotion and ask: just how efficiently can we implement fully model-free deep RL algorithms?

Perhaps surprisingly, we find that with careful design decisions in the task setup and algorithm implementation, it is possible for a quadrupedal robot to learn to walk from scratch in under 20 minutes across a range of environments. This does not require novel algorithmic components or any other unexpected innovation, but rather careful implementation of one of several existing algorithmic frameworks (and indeed multiple algorithms can work well), combined with modern optimized deep learning packages and a number of careful design decisions for the MDP formulation of the locomotion task. This result runs counter to the principles articulated in several prior works, which suggest either that simulated training is critical for quadrupedal locomotion because the training times are too long [33, 53–55, 62], that demonstration data is needed to overcome local optima challenges [54, 57], or that more sophisticated algorithmic frameworks with model-based RL [69, 162] or hand-designed movement primitives [47, 48, 50, 69, 70] are necessary for real-world training.

Our main contribution is an empirical demonstration that current deep RL methods can effectively learn quadrupedal locomotion directly in the real world in under 20 minutes. While our results largely build on existing methods, we demonstrate for the first time that a careful combination of existing components can enable direct real-world learning of locomotion skills with drastically lower training periods than reported in prior work. Our evaluation includes real-world training in four different locations (one indoor, three outdoor, see Figure 4.1), and detailed simulated analysis to understand the relative importance of different design choices.

4.2 Related Work

Roboticists have traditionally designed controllers for quadrupedal locomotion using a combination of footstep planning, trajectory optimization, and model-predictive control

	Experimental Design				Training Statistics			
					Simulation		Real World	
	Hardware	Actions	Resets	Terrains	Samples	Hours	Samples	Hours
Ours	A1	PD targets	Learned	In/Outdoor	0	0	$20 \cdot 10^3$	0.33
Wu et al. [162]	A1	PD targets	None	Indoor	0	0	$72 \cdot 10^3$	1
Smith et al. [36]	A1	PD targets	Learned	In/Outdoor	10^6	N/A	$22.5 \cdot 10^3$	1
Kumar et al. [34]	A1	PD targets	N/A	In/Outdoor	$1.2 \cdot 10^9$	24	0	0
Rudin et al. [33]	ANYmal	PD targets	N/A	Indoor	$14.7 \cdot 10^6$	0.33	0	0
Lee et al. [32]	ANYmal	PMTG [30] parameters	N/A	In/Outdoor	N/A	16	0	0
Ha et al. [27]	Minitaur	PD targets	Engineered	Indoor	0	0	$60 \cdot 10^3$	1.5
Haarnoja et al. [56]	Minitaur	PD targets	Manual	Indoor	0	0	$160 \cdot 10^3$	2
Yang et al. [69]	Minitaur	PMTG [30] parameters	Unknown	Indoor	0	0	$45 \cdot 10^3$	0.167

Table 4.1: Overview of experimental details (hardware platform used, the kinds of actions, and access to resets) and data requirements of the works most relevant—in terms of setup and task—to ours (ordered chronologically). We list the approximate numbers reported for the tasks that most closely resemble ours (walking forward). For "Training Statistics" we list the data used for *training*, i.e., we do not include data used in evaluation (that may be used in few-shot adaptation [34]), and the wall-clock time associated with collecting that data either in simulation or in the real world. Some earlier works demonstrated learning in the real world on the Minitaur [163], a relatively simple quadrupedal robot with neither shoulder nor hip abduction, in controlled, lab settings. RL with more complex quadrupedal robots [46] often used large amounts of simulation data. More similar in physical capabilities to the ANYmal and in accessibility to the Minitaur, the A1 robot has also been used to study real-world deployment in recent works.

(MPC) [44, 46, 59, 60]. Learning provides an appealing alternative to classic model-based methods, as it avoids the need for intricate modeling and extensive domain knowledge about the locomotion task, instead allowing the learning algorithm to discover a gait that works well for a given robot. Furthermore, learning in the real world allows the robot to improve over time with "on-the-job" experience.

A major question in this line of work is whether RL methods could ever be efficient enough to learn dynamic locomotion in diverse real-world environments with more complex robots. Indeed, some works have sought to sidestep this challenge by using simulation to learn controllers that are directly transferred to the real world [14, 28, 32, 33, 61–65, 164] or learning policies that can use small amounts of data when deployed in the real world to search among the space of policies it has for a suitable one [29, 31, 34, 57, 61, 66, 67]. Still, these methods rely entirely on the strategies the robot was able to learn while training in simulation. So, the training conditions must be designed and implemented such that the training captures the behaviors that are transferable to the real-world conditions. While this approach to learning controllers is often sufficient, it is non-trivial to foresee all the possible situations the robot may encounter when deployed in the real world and engineer the appropriate training environments in simulation to prepare for them. Furthermore, these learned models are fundamentally limited to those training experiences—they cannot perfectly generalize when they are tested in situations that

differ enough from their training experience, but these unexpected situations are likely to arise in the real world that is highly complex, unstructured, and unpredictable. Early work in learning *directly* in the real world explored utilizing higher level action spaces [47, 48, 50, 69, 70, 165, 166] or fine-tuning [36] to do real-world training. We show that careful implementation of already known model-free RL techniques can enable learning to walk from scratch using low-level control, such as PD targets, in several real-world environments. Next, we detail prior work in using model-free RL for real-world training of locomotion policies, followed by model-based work, and finally, systems aspects relevant to contextualizing these approaches. For a tabular overview comparing the assumptions, requirements, and results of the works most relevant to ours, see Table 4.1.

Model-free learning Much of the early work on learning from scratch in the real world applied model-free policy gradient methods to modulate pre-defined motions on relatively simple hardware. Kohl et al. [47] learn parameters of a pre-defined open-loop trajectory generator to control a Sony AIBO quadruped, and Tedrake et al. [49] learn a feedback controller to improve the control of a passive dynamic walker whose design is such that it walks when control is disabled, using policy gradient methods. Luck et al. [165] achieved sample-efficient learning by leveraging periodicity to similarly learn few parameters to tweak a finned robot's pre-defined controller. Choi et al. [70] also use a stochastic policy gradient method to learn to control a Snapbot by learning a distribution over Gaussian random paths that define fixed length joint trajectories that are then realized by an open-loop controller. Yang et al. [166] learns a model-free RL policy in the real world which outputs gait parameters and foot placements that are realized by a low-level MPC controller. While very effective for learning walking motions in the real world by shaping exploration and ensuring safety, using high-level action spaces limits the types of skills that can be learned. Recent works [27, 56] have utilized off-policy methods [72] to perform sample-efficient learning on a Minitaur (2 active DOF per leg) by directly outputting target joint positions. Training requires roughly 2 hours (160k control steps) to learn to walk forward and 1.5 hours (60k control steps per direction) to learn to walk forward and backward, respectively. Our work also uses off-policy model-free RL to learn from scratch in the real world—uniquely, though, we train an A1 quadrupedal robot (3 active DOF per leg) not only in lab settings, but also on outdoor irregular terrains, in just 20 minutes.

Model-based learning Model-based RL methods have been applied to enable an A1 robot to learn how to walk on flat ground [69, 162]. Yang et al. [69] use trajectory generators [30], which output smooth, periodic leg trajectories and use a model-based method to modulate these trajectories. Concurrent work by Wu et al. [162] uses a latent dynamics model to generate additional training data in order to reduce the burden on collecting real-world samples [167] in order to learn to walk with low-level PD targets as actions within an hour. In contrast, we use a learned policy to automatically provide

resets, and our robot learns to walk within 20 minutes in five environments, three of which are natural outdoor terrains, using a simple model-free method.

Systems considerations RL is known to require lots of data to learn complex behaviors. For example, a state-of-the-art blind quadrupedal locomotion policy trained completely in simulation required 12 hours [32]; however, real-world data is significantly more expensive to collect. Some have sought to eliminate the data collection bottleneck, using upwards of thousands of workers to collect experience simultaneously and consolidating the information for policy updates [168–170]. In the locomotion domain, Rudin et al. [33] use this parallelism to train a simulated ANYmal quadruped to walk on uneven terrain—in 20 minutes of wall-clock time—and then deploy the learned policy in the real world. However, massively parallel data collection is not often feasible in the real world. As such, the focus of another line of work has been on enabling sample-efficient methods with asynchronous pipelines. Due to computational costs, many prior works claim asynchronous training to be necessary for real-world training [56, 72, 162, 171, 172]. Most similar in spirit to our work, Haarnoja et al. [56] have a three-part asynchronous training pipeline, with jobs dedicated to data collection, motion capture for state and reward estimation. Follow-up work [27] achieved superior sample efficiency (approximately half the samples required) with the same underlying algorithm by performing synchronous training at a per-step basis (as opposed to episodic, asynchronous training), thereby reducing overall wall-clock time by a similar factor. Recently, more efficient model-free methods have been enabled by placing a larger burden on the computation. In our work, we find that we are able to support synchronous, per-step training in our implementation (see Subsection 4.3.2), achieving learning in 20 minutes on a real robot using a single GPU laptop.

4.3 Fast and Simple Real-World RL

In this section, we describe the algorithmic framework we use, which is based on standard Q-function actor-critic methods [173], building most directly on DroQ [174], which extends the SAC algorithm [72] with Dropout [175] and Layer Normalization [176]. We emphasize that our result is not enabled so much by any one algorithmic component (though the algorithm is also important!), but rather careful implementation and task setup, which we discuss in Section 4.4.

4.3.1 Preliminaries

Learning to walk can be formalized as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, p_0, p, r, \gamma)$ where $\mathcal{S} \subset \mathbb{R}^n$ is the state space, $\mathcal{A} \subset \mathbb{R}^m$ is the action space, $p_0(\cdot)$ is the initial state distribution, $p(\cdot|s, a)$ governs dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines

rewards, and $\gamma \in [0, 1)$ is the discount factor. The goal of RL is to optimize the expected discounted cumulative return induced by the policy $\pi : \mathcal{S} \to \mathcal{A}$:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].$$

We consider actor-critic methods consisting of interleaved policy evaluation and improvement steps. During policy improvement, we fit a critic to estimate the discounted returns of the current training policy starting at state s and action a:

$$J(\theta) = \mathop{\mathbb{E}}_{(s,a,s')\sim D}[(Q_\theta(s, a) - y(s, a, s'))^2]$$
$$y(s, a, s') = r(s, a) + \gamma Q_{\theta'}(s', a') \text{ where } a' \sim \pi_\eta(\cdot|s')$$

where $Q_{\theta'}$ is a target network with weights updated via exponential moving average. Then, the critic is used to improve the policy by maximizing the objective:

$$J(\eta) = \mathop{\mathbb{E}}_{\substack{s\sim D \\ a\sim\pi_\eta(\cdot|s)}} [Q_\theta(s, a)].$$

The most widely-used off-policy RL algorithms for continuous control, such as SAC and DDPG [72, 177], are commonly trained by making one critic update and one policy update after each environment step. Since the update time can exceed what is allowed by the robot's control frequency, sometimes real-world training is performed by collecting an entire trajectory and then updating the critic and policy for a number of steps equal to the length of the trajectory [56]. Next we will discuss the simple changes to these common practices we make that has enabled our result.

4.3.2 Efficient Model-Free RL

Actor-critic methods have recently become significantly more sample-efficient by improving critic training, thereby allowing more learning on the same amount of training data. For example, REDQ [58] extends SAC [72] by utilizing a large ensemble of critics and randomly sampling subsets of them. DroQ [174] regularizes the critic with Dropout [175] and Layer Normalization [176]. The common thread is some sort of regularization or normalization (or both) that mitigates the tendency of off-policy, bootstrapped critic updates to overfit and lead to overestimation [58, 174]. These techniques allow for taking significantly more gradient steps on the critic (up to 20) after each environment step, which in turn leads to significantly more sample-efficient learning. As we will discuss in our analysis in Section 4.6, a variety of approaches lead to the critical jump in improvement over the baseline method. These results suggest that the key is not any one particular choice but the general principle of augmenting actor-critic RL with regularization or normalization.

While these methods speed up learning with respect to the number of samples collected, the increased computational cost actually supersedes data collection as the primary bottleneck for real-world training. That is, a naïve implementation cannot train as fast as the samples are collected. Prior work has addressed this either by performing asynchronous training [56, 162] or training in-between trials [36]. Both add a delay between the agent interacting with the environment and learning from the samples, which slows down training. Our choice of algorithm and implementation is aimed at enabling real-time synchronous training, which we expand on in Section 4.6. For implementation, we use JAX [178], a modern machine learning framework that performs just-in-time compilation to optimize execution significantly (see Section 4.5 for a discussion of the practical aspects).

4.4 System Design

We design our system prioritizing fast training in unstructured real-world environments. We use a relatively low-level action space—rather than using pre-defined motion primitives as discussed is used in many prior works (see Section 4.2)—and use only proprioceptive information—so as to be able to train anywhere, without an instrumented motion capture system. Of course, the MDP definition and implementation often has a large impact on learning, e.g., we found the design of the action space to be particularly important. In the remainder of this section, we detail these design decisions. For our robot platform, we use the A1 robot from Unitree and build our simulation using MuJoCo [179] and DM Control [180]. The policy π_i and Q-function $\{Q_{\theta_i}\}_{i=1}^{N_{\text{ensemble}}}$ are modeled using separate fully-connected neural networks that are constructed and trained using JAX [178].

4.4.1 State and Action Spaces

For learning locomotion controllers, the robot's position is used in order to provide reward supervision [180], and privileged information is often used to train policies in simulation [32, 34]. As such, such policies cannot trivially be further trained in the real world. When training in the real world, motion capture systems have been used to localize the robot [27, 56]. However, in order to train in the wild, the robot must be able to receive feedback purely from its onboard sensors. In terms of actions, for generality, we parameterize the policy to directly output joint targets rather than rely on pre-defined gaits or trajectory generators.

The state $\mathbf{s}_t$ contains the root orientation, root angular velocity, root linear velocity, joint angles, joint velocities, binary foot contacts, and the previous action. For the root orientation, we include the roll and pitch. For angular velocity, we include roll, pitch, and yaw information (as to penalize excessive turning). We use an onboard linear velocity estimator which combines integrated acceleration and leg velocity estimated with forward

kinematics via Kalman filter that was shown to suffice for reward supervision in outdoor environments [36]. Actions $\mathbf{a}_t$ are PD position targets for each of the 12 joints and applied at a frequency of 20Hz. We define the action space for every leg as $[p - o, p + o]$ where p corresponds to initial robot pose and o is an action offset. Following Fu et al. [181], we use $o = [0.2, 0.4, 0.4]$. We confirm that constraining the action space is crucial for training the agent with RL in Section 4.6. Moreover, we found that the initial robot pose also impacts exploration significantly. In the simulator, we used $p = [0.05, 0.7, -1.4]$; however, during the early experiments in the real world, we found that $p = [0.05, 0.9, -1.8]$ promotes safer exploration on the robot since this configuration leads to fewer failures. We use a position controller where the torques commanded to the robot are computed as $\tau = K_p(q^* - q) - K_d \cdot \dot{q}$. In this case, q^* and q define the desired and current positions correspondingly, while $\dot{q}$ defined motor velocities. K_p and K_d define motor gains and damping.

4.4.2 Reward Function

Prior works have reported using very complex reward functions consisting of upwards of tens of terms to give rise to a desired motion [32, 57, 77, 181]. We found that with the state and action space choices described above, even a simple reward function was sufficient to produce naturalistic gaits. We define the reward function following standard DM Control [180] locomotion tasks, where the agent is provided with a constant reward within a target velocity interval, outside of which the reward linearly decays to 0.

$$r(s, a) = r_v(s, a) - 0.1 v_{yaw}^2$$

where v_{yaw} is an angular yaw velocity and

$$r_v(s, a) = \begin{cases} 1, & \text{for } v_x \in [v_t, 2v_t] \\ 0, & \text{for } v_x \in (-\infty, -v_t] \cup [4v_t, \infty) \\ 1 - \frac{|v_x - v_t|}{2v_t}, & \text{otherwise.} \end{cases}$$

where v_t is the target velocity, while v_x is a forward linear velocity in the robot frame. Our choice of the reward function is motivated by simplicity.

During early experiments with the real robot, we found that using the forward velocity in the robot's local frame caused it to dive forward as falling quickly onto its head does correspond to high forward velocity from the robot's perspective. However, our goal is for the robot to move forward in the global frame, parallel to the ground plane, so we instead project the forward velocity onto the ground plane. To keep the robot from leaving the training area with minimal interruption, we simply continue training while providing a reward of 0 when we detect that someone has lifted the robot to move it (i.e. when there

Figure 4.2: **(Real-world test terrains)** Examples of learned gaits acquired on a variety of real-world terrains. Left to right: flat, solid ground covered in dense foam mats; a 5cm memory foam mattress; loose ground comprised of eucalyptus bark; a grassy lawn; a gently sloped hiking trail.

are no foot contacts detected). Otherwise, the robot trains continuously, terminating only when the robot's roll or pitch exceeds 30 degrees. To reset the robot in the real world, we use the open-source reset policy from Smith et al. [36].

4.5 Learning in the Real World

We aim to answer the following through our experiments:

(1) How quickly and consistently can the robot learn to walk in the real world using model-free RL?

(2) Can this approach enable a robot to learn to walk not only on flat ground in the lab, but also 'in the wild'?

In order to understand the variance across random seeds and small changes in real-world conditions (e.g., the status of the hardware, how many times the robot was redirected, etc.), for one of our experiments, we train the robot four times in the same controlled environment. To study (2), we train the robot in four additional terrains, three of which are outdoors. Lastly, we discuss our findings from using the design decisions, tuned in simulation as described in Section 4.4, to train in the real world. Videos of all real-world training along with code to reproduce our results can be found at `https://sites.google.com/berkeley.edu/walk-in-the-park`.

4.5.1 Setup

We conduct experiments with five terrains (see Figure 4.2), each of which possesses unique characteristics:

(1) *Flat, Solid Ground:* We placed tiles of dense foam on the floor for protection that make for a high-friction surface.

(2) *Memory Foam Mattress:* The other indoor terrain we test on is a 5cm-thick memory foam mattress. The robot's feet sink fairly deep into the surface—see, e.g., the depression in the surface of the mattress made by the robot in the earliest frame in Figure 4.2 (second from left), making walking difficult and requiring a unique gait to attain ground clearance.

(3) *Mulch:* A layer of bark (about a foot deep) makes walking especially difficult as the terrain is not only heavily irregular but also obstructive. As shown in Figure 4.2 (middle), the robot naturally sinks such that the lower half of its legs are submerged in the bark. Thus, it must learn to churn through it in order to move.

(4) *Lawn:* The lawn presents a lower friction, cushioned, deformable walking surface.

(5) *Hiking Trail:* Here the surface material is a compact, dry dirt. As shown in Figure 4.2 (right), the trail is at a slight incline and irregular (tree roots, pebbles, troughs, etc.), presenting additional challenges.

To be able to train outdoors, we use a laptop (Origin EON15-X) for training equipped with a single NVIDIA GeForce RTX 2070 GPU. For all experiments, we collect data for about 1 minute (1000 time-steps) by sampling from the action space uniformly at random. We then train synchronously, making a policy update (20 critic updates and an actor update) in between each action executed on the robot every 0.05 seconds. On a standard workstation equipped with a NVIDIA GeForce RTX 2070 GPU, our initial JAX implementation of DroQ was capable of performing 700 critic updates per second, roughly corresponding to 35 time-steps per second. However, this was insufficient for training the agent on a laptop, which reiterates the importance of implementation for real-world training. For our final implementation, we jit 20 critic updates corresponding to one environment step, resulting in 2400 critic updates per second which corresponds to running training at 120 Hz. In contrast, for a baseline comparison, our best PyTorch implementation was not fast enough for synchronous training, only permitting an effective control frequency of 7.5 Hz.

4.5.2 Results

We report the average velocity over intervals of 1000 time-steps (corresponding to 1 minute of wall-clock time) during training in Figure 4.3 with respect to the number of real-world samples collected during which the robot was on the ground (not including the times the robot was lifted and reoriented). Per answering (1), we report the average and standard deviation (solid line and shaded region, respectively) across four runs on the flat, solid ground (pink). In all cases, the robot learns to walk in less than 20,000 samples (roughly 17 minutes' worth of data which amounts to 20 total minutes of wall-clock time due to various minor sources of overhead: jitting (about 1 minute), resetting and reorienting the

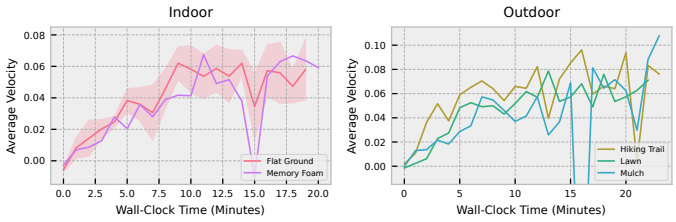


Figure 4.3: **(Real-world learning from scratch results)** Learning curves for all the real-world experiments showing the average velocity of the stochastic policy with respect to real-world, wall-clock time. Note that the robot runs continuously, training in between sending actions to the physical hardware. 1000 time-steps corresponds roughly to 1 minute of training time, all things considered.

robot, etc.). At the start of training, the robot mostly shuffles, slowly edging backwards. In 5 minutes, the robot learns to inch forward but is unstable. Within 10 minutes, the robot learns to take fairly large steps, but it has not learned to maintain balance while taking these larger steps. Finally, after 15 minutes of training, the robot adopts more conservative behavior in order to both walk and remain balanced.

Next, we train on the variety of challenging terrains detailed in Subsection 4.5.1. On the memory foam, the robot makes its way out of the initial position it had sunk into during initial data collection (similar to the situation shown in the earliest frame of Figure 4.2 (second from left)) within about 5 minutes. In contrast to the flat, solid ground, the robot falls much less frequently, but its feet often catch on the fabric cover and it needs to learn to walk in a way such that it avoids dragging the fabric with it in order to traverse it. On the thick layer of wood mulch, the initial data collection essentially digs the robot quite deep into the loose ground. In the first ten minutes, the robot learns to kick up its front feet in order to dig itself through, as seen most clearly in the latest frame of Figure 4.2 (middle) where the robot is kicking its front, right leg up to pull itself forward. Furthermore, the terrain is irregular, yet it is able to quickly adapt its behavior. In all cases the robot learns to traverse each given terrain with roughly 20,000 samples (20 minutes).

4.6 Analysis of Design Decisions

This section presents simulated comparisons of design decisions and SAC variants we considered in this chapter. Since our goal is to run training on a real robot, we aim for design decisions and algorithms that lead to improved stability and sample efficiency. To match the real-world setup, we simulate the official A1 model in MuJoCo, and used the same position controller and rewards as discussed in Section 4.3.2. We provide the exact model we used in the open-sourced code: `https://github.com/ikostrikov/walk_in_`

`the_park`.

MDP formulation First, we observe that the value of damping for the position controller used for the robot significantly impacts learning (see Figure 4.4a). Small damping values ($Kd = 1$) lead to instabilities, which are not desirable for a policy executed on a real robot, while large values prevent the agent from reaching the target velocity ($Kd = 20$). Therefore, for the remaining ablations, we used the value of damping set to 10. We also evaluate other design decisions in Figure 4.4b. In particular, we confirm the efficacy of constraining the action space: we observe that the simulated agent cannot make any progress in the unconstrained action space, while constraining the space leads to stable training and does not prevent the agent from reaching the target velocity. Finally, we note that applying a low-pass filter to the PD targets to promote smoothness, as is common practice [27, 36, 57], degrades learning efficiency. This is perhaps unsurprising, as the filter creates a dependency on action history that violates the Markovian assumption.

Algorithms Our goal is to train a robot to walk in the real world as efficiently as possible, where efficiency includes computational complexity, sample complexity, and total wall-clock time. As discussed in Section 4.3.2, we utilize off-policy, model-free actor-critic methods (basing all on SAC [72]), and investigate a variety of regularization and normalization techniques to accelerate them. We present an extensive comparison in Figure 4.4c, with the aim of understanding which ingredients are important for attaining the requisite sample efficiency for training in the real world. Standard SAC (purple) with an update-to-data (UTD) ratio of 1 takes one gradient step on the critic for every one time step of data collection. Efficiency can be increased by taking more gradient steps, and we show standard SAC with a larger UTD ratio of 20 (dark blue) as well. We see that naïvely increasing the number of critic updates made per time-step improves sample efficiency, but still requires roughly 30k samples, which would amount to roughly 30 minutes' worth of data, to reach the target velocity. We implement REDQ [58] by modifying SAC with UTD ratio of 20 to use random subsets of a larger ensemble of networks in order to calculate target values, and we see this regularization indeed leads to improvement (roughly 5k fewer time-steps, or 5 minutes wall-clock time, required). As noted in DroQ [174], REDQ is more computationally expensive due to the large ensemble, and regularizing the critic with a combination of LayerNorm and Dropout can lead to similar benefits at lower compute cost (blue). However, we also consider LayerNorm (pink) and Dropout (orange) in isolation (each also with a UTD ratio of 20). Perhaps surprisingly, Dropout alone already performs similarly to REDQ, whereas LayerNorm (even without Dropout) leads to even better performance.

We conclude that a variety of regularization or normalization methods, if implemented and applied carefully, can all achieve a similar level of improvement in performance over their underlying algorithm *in our setup*. That is, the important thing is not any single specific

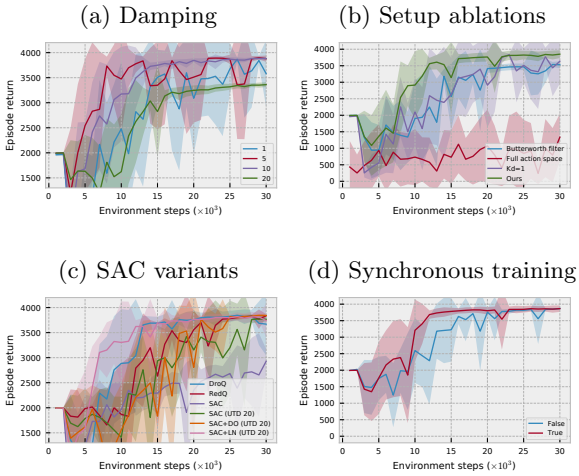


Figure 4.4: **(Empirical analysis of design choices in simulation)** Experimental evaluation of: (a) performance for different values of the damping parameter for the position PD controller; (b) ablations of various task setup choices; (c) the effect of the frequency of policy updates (between time-steps versus episodes); (d) regularization and normalization methods for efficient RL. Each curve and shaded region represents the average and standard deviation across 10 random seeds.

regularization technique, but the use of any suitable regularization so as to enable SAC to effectively use higher UTD ratios. We also compare updating the agent between episodes and after every environment step and notice that getting immediate feedback leads to more stable training and faster convergence (see Figure 4.4d). In order to facilitate this kind of training synchronously, the updates must be inexpensive enough to be able to perform them between time-steps (of which there are 20 per second). As such, we favor using the less computationally expensive DroQ variants over others in the real world.

4.7 Conclusion

We present our finding that we can train quadrupedal robots to walk via deep RL in real-world settings, such as grass, loose ground, forest trails, and mattresses, with about 20 minutes of training. We demonstrate that this can be enabled using a high-quality implementation of existing algorithmic ideas combining standard actor-critic algorithms with one of a number of regularization strategies. We compare the different design choices in simulation and show that a variety of designs can work well, and then demonstrate in the real world that this leads to highly efficient and successful learning on a range of different terrains. Our empirical results show that real-world training of locomotion policies via RL can be significantly more practical than previously believed, and does

not necessarily require significant deviations from existing practice in RL, but rather careful combination of current best practices. We hope that our work will serve to further encourage investigation of real-world RL in robotics.

Chapter 5

Continuous Improvement with Real-World RL

5.1 Motivation

The real world is noisy, diverse, unpredictable and challenging. To confront this complexity, a robot must be capable of versatile and robust behaviors. Designing controllers that anticipate and handle *any* scenario a robot will encounter in its lifetime, whether through manual engineering or learning, is impractical. An alternative to providing a robot with predetermined capabilities is to instead equip it with the ability to autonomously improve during deployment, learning from its mistakes as it encounters unforeseen circumstances. Legged robots in particular have incredible mobility and can reach places challenging or unsuitable for humans, for example, in search and rescue scenarios, and thus must be robust to these unforeseen situations.

Reinforcement learning (RL) offers a general framework for such a 'self-improving robot,' providing a data-driven approach for learning behaviors through interaction. However, the practical application of end-to-end RL in robotic systems is often not straightforward [182, 183], with many of the challenges stemming from high sample complexity: RL algorithms are notoriously data-hungry, while real-world data collection is notoriously expensive due to human supervision requirements, hardware damage, and other physical constraints. Although recent works [26, 47, 109, 151, 152, 154, 155, 184] have demonstrated encouraging progress toward end-to-end RL on real-world systems, often by applying the latest advances in sample-efficient RL, the efficiency and final performance of these methods still presents difficulties for persistent and reliable deployment on real-world platforms such as legged robots. In this chapter, we consider the task of learning quadrupedal locomotion and ask: how can we enable a robot to learn more agile locomotion in the real world, where it must learn and adapt efficiently amid diverse, challenging

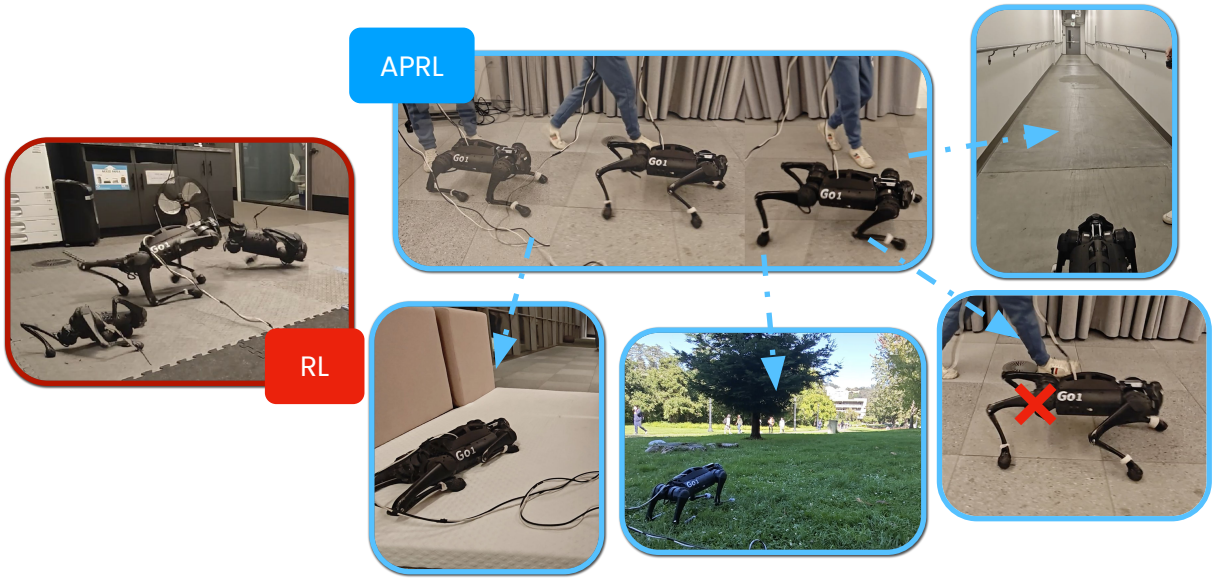


Figure 5.1: **(Overview of APRL)** APRL uses a novel action space regularization technique based on dynamics prediction error to modulate exploration over the course of training. This enables real-world quadrupedal learning that can traverse challenging terrains and continually adapt to changes in dynamics.

scenarios?

We present APRL, a system that addresses the real-world challenges of *efficiency* and *continual improvement* in robot learning via adaptive policy regularization, with a focus on quadrupedal locomotion. Our key observation is that the policy's search space of actions has a significant effect on the robot's ability to learn effectively. To illustrate this, consider tasking the robot to learn to walk with no prior knowledge. With full command of its joint range, most random behaviors will lead to catastrophic failure, and exploration becomes practically infeasible (see the behavior pictured on the left in Figure 5.1), but manually defining an appropriate space of actions to explore restricts the robot's capabilities, as the highest performance later in training might necessitate exceeding such restrictions. Our approach is to *dynamically regularize* the policy, providing it with enough freedom to explore and improve, but not so much that its exploration is needlessly expensive. At the start of training, APRL biases the policy toward small-magnitude actions to avoid the robot having to first learn this through costly first-hand experience, but as the robot becomes more competent, it should be allowed to explore more aggressive actions. To this end, we introduce an adaptive penalty based on how *familiar* the current situation is for the robot. This adaptive strategy allows the robot to explore more aggressively once it has learned about its surroundings, and dial back its exploration if it encounters unexpected dynamics, where we would expect the policy to not generalize well. We

measure familiarity by training a dynamics model on the data the robot has collected and using its prediction error to dictate the policy's recommended search space. We use this action regularization synergistically with resetting the agent [185], which combats the early overfitting, a common problem sample-efficient RL algorithms are prone to. Resets improve plasticity, providing more opportunities to learn when needed, and the dynamic penalty on actions focuses the robot's behavior, i.e., to prevent excessive falling and inefficient exploration.

In summary, this chapter presents a system, APRL, for efficiently learning quadrupedal locomotion directly in the real world using a novel, adaptive regularization strategy that promotes more efficient, high-performing, and stable training. We demonstrate that on a 12 degree-of-freedom Unitree Go1 quadruped, our system can learn to walk forward in just minutes starting completely from scratch. While prior work saturates in performance, APRL allows the robot to continuously improve with more training. Furthermore, the behavior learned with APRL performs significantly better (on average by a factor of 2) in terms of its average walking speed in challenging situations as shown in Figure 5.1, such as on an incline, on a memory foam mattress, and through thick grass, and can be fine-tuned in each to further improve performance. Our code to reproduce all results (including a real-world environment and training) is available on our website: `https://sites.google.com/berkeley.edu/aprl`

5.2 Related Work

Legged locomotion has long been of interest to roboticists, and a large body of work is dedicated to developing controllers by means of model-based optimal control [44, 46, 59, 186–189]. Model-based methods have enabled a range of robotic locomotion skills, from high-speed running [60] to backflipping [190], but require careful modeling of the conditions for which they will perform well in. Recent research has also achieved remarkable success by using learned approaches. One such approach that bypasses the complexities of real-world learning leverages simulation to train behaviors that transfer to the real world, demonstrating impressive results from robust walking [28–34, 57, 61, 62, 66, 164] to more complex behaviors like agile running [191], jumping [192, 193] or bipedalism [41, 194–196]. These approaches rely on zero-shot generalization, which has been shown to suffice in many scenarios of interest. However, they have two key limitations: they require extensive engineering of the simulation settings for the policy to generalize, and most do not have any mechanism to learn from their failures in the real world.

The alternative learning approach we explore in this chapter is to directly learn in the environment of interest. Early work on learning *directly* in the real world explored utilizing higher level actions in trajectory space [47, 48, 50, 69, 70, 165], limiting their applicability to skills beyond walking or running at different speeds. Most similar to our setup are works

that have studied real-world training using low-level PD target actions, with a 12-DoF A1 robot enabled by sample-efficient RL. Wu et al. [162] report learning to walk without resets and to recover from pushes in 1 hour using a model-based RL algorithm. We build our system on the simple model-free framework introduced by Smith et al. [38], which demonstrated learning to walk from scratch in various environments in under 20,minutes. While extremely efficient, the maximum achieved velocity in this prior work was less than 0.1 m/s, and the method did not demonstrate transfer between or adaptation to different terrains. Our work differs in three ways: (1) it leads to a **1.4x** improvement in the speed achieved after continued training; (2) it enables the resulting policy to perform better when directly transferred to more challenging situations; and (3) it demonstrates rapid adaptation through continued training when the policy does not generalize well in unseen settings.

A component of our approach is to regularize the actions produced by the policy according to model misprediction, which leads to fewer falls and lower torques during training. For locomotion, falls lead to physical damage and increased wall-clock training time. Ha et al. [27] make this observation and use a constrained MDP (CMDP) formulation to explicitly combat falling during training by penalizing actions that lead to such states. We find in Section 5.6 that the CMDP approach still requires many failures in order to properly assign credit to which actions lead to fallen states. Our action regularization is similar insofar as it manifests as a penalty for the actor, but we directly regulate the magnitude of actions the policy explores within. Therefore, in APRL, the policy does not need to first learn which actions lead to falls by taking them in the real world and experiencing failure. We leverage our knowledge that small actions around the nominal pose are unlikely to cause failure, and bias the policy to explore in this space first. While this strategy empirically leads to fewer falls than fully unbridled exploration, our primary goal is not to formally ensure safety but to enable efficient learning in the real world.

5.3 System Design

We use the Markov Decision Process (MDP) formalism to define our task of learning to walk. An MDP is defined by a state space $\mathcal{S} \subset \mathbb{R}^n$, action space $\mathcal{A} \subset \mathbb{R}^m$, initial state distribution $p_0(\cdot)$, transition function $p(\cdot|s,a)$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0,1)$. RL maximizes the expected discounted cumulative return induced by the policy $\pi : \mathcal{S} \to \mathcal{A}$:

$$\mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].$$

We use a type of algorithm that fits a critic $Q_\theta(\mathbf{s}, \mathbf{a})$ to estimate this objective, and the policy then uses the critic to improve by maximizing the objective:

$$\mathcal{L}_{\texttt{act}}^{\texttt{SAC}}(\phi) = \mathop{\mathbb{E}}_{\substack{s\sim D \\ a\sim\pi_\phi(\cdot|s)}} \Big[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \Big].$$

We build on the actor-critic RL method and implementation used by Smith et al. [38] (details in *(b)*). We additionally use resets to improve plasticity [185, 197]. All neural networks are 2-layer feed-forward networks constructed and trained using JAX [178]. We use an Origin EON15-X laptop with an NVIDIA GeForce RTX 2070 GPU.

MDP formulation We use the Go1 quadruped from Unitree for real-world experiments, and perform analysis in simulation using the MuJoCo Menagerie model [198]. The robot's task is to walk as fast as possible without falling, where falling is defined as the robot's roll or pitch exceeding 30 degrees from upright. The robot's state $\mathbf{s}$ comprises its root orientation, joint angles and velocities, root (local) velocity, normalized foot contacts, and last recorded action. Previous works [38, 162] used a Kalman filter to fuse forward kinematics and acceleration to estimate the robot's velocity. In this work, we instead obtain velocity estimates from an Intel RealSense T265 camera attached to the robot's neck. We found the vision-based estimator to be more reliable and less susceptible to drift. We use target joint angles for the action space as is common for learned locomotion controllers on similar hardware [34, 57, 181]. The control frequency is 20Hz. The target joint angles are fed to a PD controller, with a position gain of 20 and derivative gain of 1, which converts them to torques at the leg joints. We also employ two forms of smoothing: a low-pass filter on the policy outputs and action interpolation, which is running at 500Hz. We define a reward function to maximize the robot's local linear velocity while maintaining an upright orientation. We also include penalties on angular velocity and torque smoothness.

Replay ratio and resets Sample-efficient RL methods use a high *replay ratio*, i.e., the ratio of gradient updates to real-world transitions collected, to make efficient use of the collected data. To be able to take more updates on the same data, they require some form of regularization, e.g., using model-generated data [167, 199], weight-based regularization [174, 200], ensembling at the agent level [201] or at the critic level [58], or a combination of techniques [202]. APRL uses a high replay ratio with a model-free method, using Dropout [175] and LayerNorm [176] to regularize the critic. Nikishin et al. [185] showed that excessive training on early data with high replay ratio methods can cause the networks to lose plasticity, the ability to continue learning with more data, and propose periodic resets of the agent to mitigate this effect. Resetting specifically implies reinitialization of network weights and optimizer states while maintaining the

Algorithm 5 `APRL` pseudocode

Require: Action regularization configs: growth rate N_{curr}, initial range $\mathcal{A}_{\text{start}}$, end range $\mathcal{A}_{\text{end}}$, penalty weight σ, dynamics shift threshold Δ_M, replay ratio rr, max gradient steps ∇_M.

Initialize parameters of Q_θ and π_ϕ and a replay buffer $\mathcal{B} \leftarrow \emptyset$
Initialize action regularization states $i = 0, \mathcal{A}_{\text{i}} = \mathcal{A}_{\text{curr\_i}}, \mathcal{A}_{\text{e}} = \mathcal{A}_{\text{curr\_e}}$
repeat
    Collect transition $(\mathbf{s}_t, \mathbf{a}, \mathbf{s}_{t+1}, r_t)$ with π_ϕ and store in $\mathcal{B}$

    // Update regularization
    Increment counter $i+ = 1$
    Determine progress $\alpha_{\text{curr}} = \text{clip}(i/N_{\text{curr}}, 0, 1)$
    Calculate corresponding space $\mathcal{A}_{\text{curr}} \leftarrow \alpha_{\text{curr}}\mathcal{A}_{\text{e}} + (1 - \alpha_{\text{curr}})\mathcal{A}_{\text{i}}$
    Calculate dynamics error $\Delta_{\text{curr}} \leftarrow \left(\mathbf{s}_{t+1} - \hat{f}_\psi(\mathbf{s}_t, \mathbf{a}_t)\right)^2$
    if $\Delta_{\text{curr}} \geq \Delta_M$ **then**
        Set $i \leftarrow 0$, $\mathcal{A}_{\text{i}} \leftarrow 0.9 \times \mathcal{A}_{\text{curr}}$

    // Perform updates
    for rr steps **do**
        Update θ with critic loss
    Update ψ with $\mathcal{L}^{\texttt{dyn}}(\psi)$ in Equation 5.2
    Update ϕ with $\mathcal{L}^{\texttt{APRL}}_{\texttt{act}}(\phi)$ in Equation 5.1

    // Periodically reset weights
    if $i \cdot rr > \nabla_M$ **then**
        Reinitialize θ, ϕ, ψ and reset $i = 0$
until forever

replay buffer. We incorporate this regularizer into our adaptive strategy as we will describe next.

5.4 Efficient Learning of Legged Locomotion with Adaptive Policy Regularization

We present our system for efficiently learning and fine-tuning quadrupedal locomotion in real-world scenarios using **A**daptive **P**olicy **R**egu**L**arization (APRL). Our framework, shown in Figure 5.2, involves dynamically modulating policy regularization over the course of training to provide the policy with adequate room to explore and improve, but not so unbridled as to lead to inefficient—and often violent—training. To do so, we introduce 'soft' constraints on the actions (defined in *(b)*) that are adjusted based on how 'familiar' the robot is in its current situation (described in *(c)*). We also incorporate resets to improve plasticity, i.e., the ability to keep learning from new data. In the remainder of this section, we describe the principle underlying our choice of regularization. We then

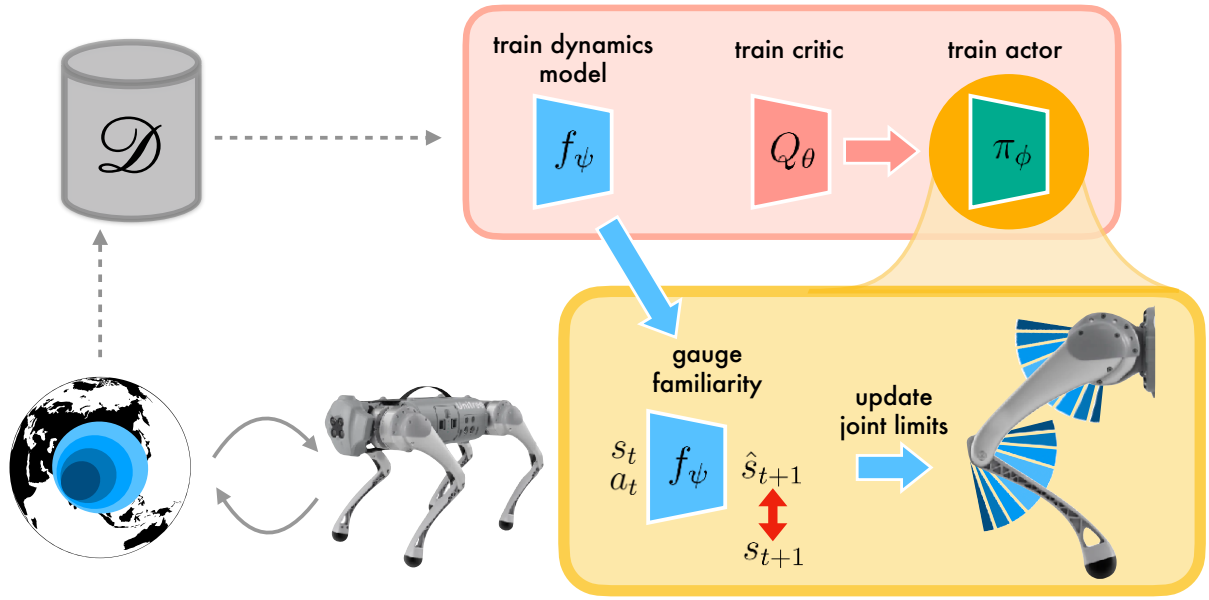


Figure 5.2: **(Depiction of APRL system)** We train the robot with a flexible constraint, represented by blue semicircles around the joints. It collects experience, storing it in a replay buffer for training an actor and critic, as explained in Section 5.3, alongside a predictive dynamics model. The model's prediction error adjusts the constraint's bounds, either tightening (for high error) or relaxing (for accurate predictions). This adjustment is incorporated into the actor's loss, as specified in Equation 5.1.

detail how we adapt the constraints based on the robot's learning progress and finally how we implement them in practice. Algorithm 5 summarizes the training procedure in pseudocode.

An efficiency-performance trade-off Prior work has shown that explicit action limits have an enormous effect on learning speed in the real world [38]—intuitively, limiting the policy's search space makes finding a solution faster. In the case of legged locomotion, where policies are often parameterized to output PD targets as actions [34, 41, 57, 181], searching in a limited region around a nominal pose is reasonable since the policy may still learn to walk at a low speed and is less prone to falling. These details are important for real-world learning, as each fall incurs nontrivial physical damage and time costs (specifically, an additional 5-10 seconds which translates to an opportunity cost of 100-200 time steps). Beyond falling, large changes in PD targets translate to large torques, which can cause significant damage to the robot over time. As we show in our experiments, while a restricted action space is helpful for learning to walk quickly, it can significantly inhibit the learned policy's ultimate capabilities.

Soft policy constraints A straightforward approach to constraining the policy's actions is to use a transformation that either maps any action beyond the limits to the corresponding extremum or scales the actions within fixed bounds. We find that these naïve implementations do not work well in practice (see Section 5.6) for our application. We instead introduce an action penalty, which can be interpreted as a soft constraint on the policy. Specifically, each action dimension is a target joint angle, where the minimum and maximum correspond to the lower and upper physical limits, respectively. We define a feasible region $\mathcal{A}_\epsilon$ that corresponds to actions whose elementwise magnitude is no greater than ϵ, which can write as an inequality constraint on each dimension i: $c_i(\mathbf{a}, \epsilon) = |\mathbf{a}_i| - \epsilon \leq 0$. Using a penalty method amounts to training with a penalty on infeasible iterates, and we found an L1 penalty with fixed weight $\sigma = 10$ to work best in our case. This leads to the modified actor loss:

$$\mathcal{L}^{\texttt{APRL}}_{\texttt{act}}(\phi) = \mathop{\mathbb{E}}_{\substack{s\sim D \\ a\sim\pi_\phi(\cdot|s)}} \Big[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \sigma \sum_i \max(0, c_i(\mathbf{a}_t)) \Big]. \tag{5.1}$$

Deciding on appropriate feasible regions As previously mentioned, we would intuitively like to allow the robot to be more exploratory when in a familiar setting. Conversely, we would like to constrain the robot to promote more conservative behavior when the robot is in a setting that is different from that in which it was trained. To do this, we use a dynamics model: we fit $\hat{p}_\psi(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ on the data the robot collects and use the likelihood of the latest observed transitions to determine whether a situation is 'familiar'. We specifically choose this heuristic over others, e.g., model disagreement to measure epistemic uncertainty [203], to implicitly encourage the policy to favor predictable solutions. Dynamics prediction error also explicitly detects changes in the environment dynamics, which we would like the robot to be able to immediately react and adapt to. We represent the dynamics model as $\hat{p}_\psi(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) = \mathcal{N}(\hat{f}_\psi(\mathbf{s}, \mathbf{a}), I)$, where f_ψ is a neural network. Training with maximum likelihood corresponds to a mean squared error (MSE) loss:

$$\mathcal{L}^{\texttt{dyn}}(\psi) = \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D} \Big[(\hat{f}_\psi(\mathbf{s}, \mathbf{a}) - \mathbf{s}')^2 \Big]. \tag{5.2}$$

We use a schedule such that the joint limits are grown at every time step by a constant increment (line 8) unless the likelihood of the most recent data is registered as highly unlikely by the learned dynamics model, i.e., if the MSE incurred by the dynamics model surpasses a maximum of Δ_M (lines 9-10). If this threshold is hit, we shrink the joint limits by a multiplicative factor (line 11) to allow the robot to react quickly in a new situation.

5.5 Real-World Results

Our real-world experiments test whether APRL can enable a real quadrupedal robot to efficiently learn to walk entirely in the real world and adapt to new dynamics that are

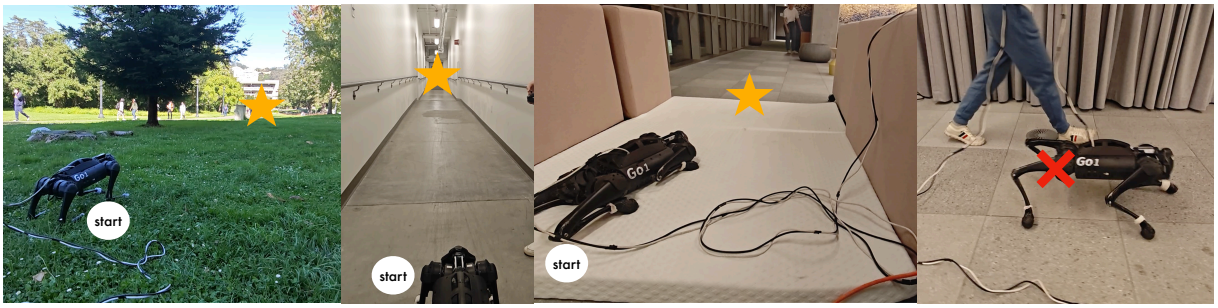


Figure 5.3: **(Test environment visualization)** We visualize the different environments we test in: Grass, Ramp, Mattress, and Frozen Joint. We indicate the start and goal locations for the situations in which there is a path for the robot to traverse and is evaluated on its time to finish the entire path.

more challenging than demonstrated by prior work. We specifically seek to answer the following questions:

1. Can we enable a real 12 DoF quadrupedal robot to learn to walk in a matter of minutes *without* a manually defined restricted action space?
2. Does APRL enable *continued improvement* as the robot collects more data?
3. Does APRL enable the robot to learn to walk in more challenging settings?
4. Can we use APRL to allow the robot to continue learning amid changing dynamics?

5.5.1 Experimental Setup

We address the first two questions by comparing to the prior work of Smith et al. [38] (labeled as *Restricted [46]*), as this prior method also focuses on learning to walk directly through real-world training, in the same, flat-ground environment training for 80k steps each (roughly 80 minutes of real-world interaction time). To address (3) and (4), we evaluate the learned policies in 4 new scenarios (shown in Figure 5.3):

- **Mattress**: The robot must walk across a 5cm thick memory foam mattress. The robot's feet sink and the depression of the mattress makes walking more difficult, requiring a unique gait with more force.
- **Ramp**: We task the robot to walk up a slippery, 5-degree inclined ramp. The inclination and slipperiness require the robot to maintain good balance while giving strong pushes on the back legs to climb up.

Figure 5.4: **(Qualitative comparison of policies)** We compare the gaits learned, (top) Restricted and (bottom) APRL, from scratch on flat ground by showing a time-lapse of the policies rolled out for 5 seconds each. Our policy learned to use its front legs to step and propel its back legs in a cantering-like manner whereas the Restricted policy drags and slides across the ground.

- **Grass**: We deploy the robot outdoors on grass. The unevenness of the mud underneath and the unique friction properties require good foot clearance and quick response to changes in the shape of the terrain.
- **Frozen Joint**: We freeze the thigh joint on the rear right leg to test adaptation to sudden shifts in dynamics in a controlled way, where there are no natural variations inn terrain to cause differences in performance.

In these evaluations, to account for stochasticity and variance in the real world, we run each evaluation at least 3 times and report the mean and standard deviation across these runs. For (3), we first test the policies without any fine-tuning. We introduce two additional evaluation metrics: the time each policy requires to traverse from one end of a path to another (pictured in Figure 5.3 as the white circle and gold star, respectively) and the number of times the robot fell while doing so. For these evaluations, we manually reset the robot onto the path if it veers too off-course. Note though that we do not include the relative finish time and fall counts for the "Frozen Joint" scenario because the shift in dynamics is not in terrain. Lastly for (4), we evaluate whether a few minutes of continued training (specifically 3k time steps) allows the robot to improve in these scenarios. That is, we fine-tune in the target setting, then re-evaluate using the same protocol.

5.5.2 Results

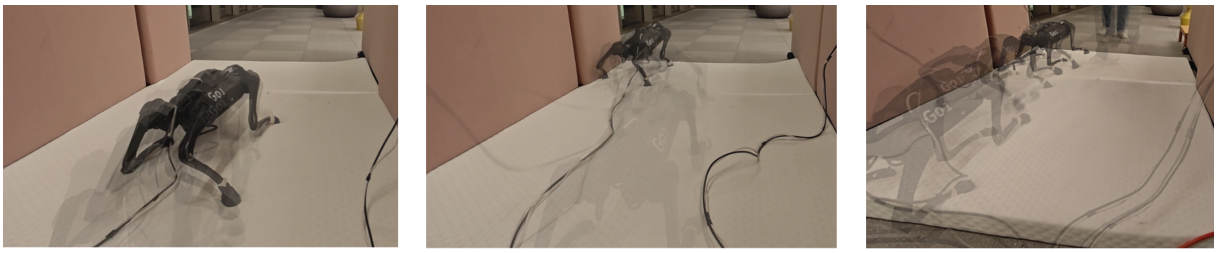


Figure 5.5: **(Qualitative comparison on new terrain)** We show a 5 second time-lapse of evaluating different policies on the mattress. The Restricted method tries to slide on the mattress, which slows it down significantly. APRL policies have a higher foot clearance, so they are able to traverse it more efficiently and, after fine-tuning, with fewer falls.

Training from scratch. In Figure 5.6), we see that even without manual restriction of the action space, APRL starts learning to walk immediately. We attempted to compare to training without regularization; however, the robot's actions were too aggressive to collect even a small amount of data. APRL's adaptive regularization makes training possible in way that, importantly, allows the robot to *continue to improve*, achieving a maximum average velocity of 0.62 m/s. In contrast, the Restricted method indeed learns to walk extremely quickly but plateaus early in training and achieves a maximum average velocity of only 0.44 m/s. This performance almost matches that of its simulated variant (see Section 5.6), so we believe this cap to be a fundamental limitation rather than a real-world-specific challenge. APRL's performance also closely tracks its simulated variant's; however, we hypothesize that our performance is limited in the real world partially due to space constraints, as the robot is only able to take a few steps before reaching the workspace limits once it reaches a higher speed. We also observe that APRL produces a visually more naturalistic gait, shown in Figure 5.4 and on the website: `https://sites.google.com/berkeley.edu/aprl`. These results show that APRL is significantly better equipped than naïve RL to continually improve as it collects more data, rather than quickly reaching but plateauing with limited capabilities.

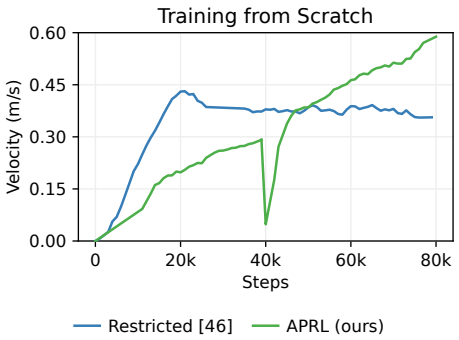


Figure 5.6: **(Forward velocity achieved during training)** The Restricted method learns more efficiently at first but is unable to keep improving. Meanwhile, our method still learns to walk quickly but acquires a maximum velocity of 0.62 m/s. Note that the dip at 40k steps is due to parameter resets to improve plasticity (line 19 in Algorithm 5).

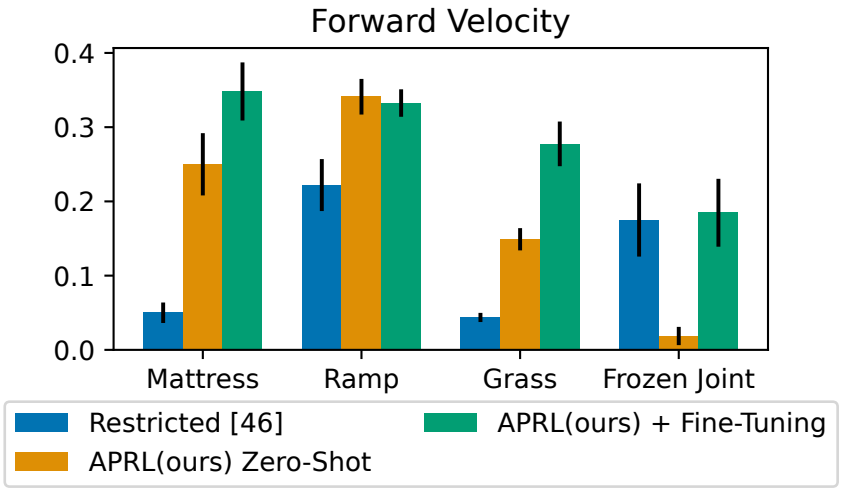


Figure 5.7: **(Real-world velocity comparisons)** In all scenarios except frozen joint, APRL significantly outperforms the Restricted method in terms of its velocity when tested in new scenarios. With just minutes of fine-tuning, APRL significantly improves performance in all settings except on the ramp, where it is comparable.

Transferring to different scenarios. We find that APRL not only successfully enables a quadrupedal robot trained only in the real world to walk amid a variety of conditions, but also to *keep improving* as it continues to be deployed. Quantitatively, the policy learned with APRL even without fine-tuning is significantly better on average at walking than the Restricted policy in terms of average velocity (see Figure 5.7) and at completing a given path faster and with fewer falls (see Figure 5.8). The exception is when we freeze a joint, in which case the Restricted policy generalizes much better during zero-shot evaluation. In this case, we find that with continued training, APRL can quickly learn to overcome this gap. In Figure 5.5, we show a qualitative comparison of policies where the path can be visualized with a static camera. We encourage the reader to view the qualitative differences in policies for each scenario at `https://sites.google.com/berkeley.edu/aprl`.

5.6 Simulated Analysis

In this section, we analyze APRL using a simulated version of the task described in Section 5.3. Although simulation does not model many of the real-world complexities that we aim to address, we use it to perform controlled experiments for comparison purposes and insight. We design our simulated experiments to answer the following questions:

1. Does restricting the action space actually cap the robot's achievable velocity?

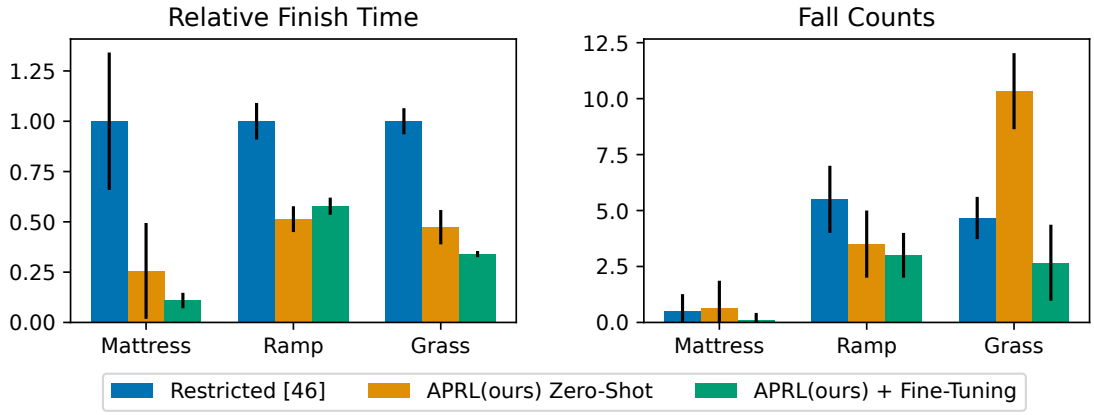


Figure 5.8: **(Real-world finish time and fall count)** (Left) We report the time taken to traverse a path relative to the Restricted method and absolute fall count. On Ramp and Grass, APRL is **2x** faster, and on Mattress, APRL is **almost 4x** faster.

2. If so, does APRL allow overcome these limits, and how does it compare to 'optimal' behavior?

3. How does APRL compare to prior work and ablations?

To answer (1), we compare learning with a fixed, limited action range as done in the Walk in the Park system [38] (labeled *Restricted*) to learning with the full action range without our adaptive regularization (labeled *No Regularization*). As mentioned in Section 5.5, this comparison is not feasible in the real world but gives the policy the most freedom to optimize, so we use its asymptotic performance as the upper bound on the robot's capabilities. We see in Figure 5.9 that the way actions are explored has a profound effect on training performance. Restricted excels at the very start of training, achieving a steeper learning curve and fewer falls than No Regularization, but also saturates very quickly and is not capable of improving beyond a velocity of about 0.5 m/s. In contrast, the policies with access to the full joint range eventually far exceed the hard-constrained policy's performance as measured by return.

For (2), we observe that APRL achieves near-optimal performance in comparing its asymptotic performance in terms of return and achieved velocity, with that of the non-regularized 'oracle'. Furthermore, it does so with significantly fewer falls, making it feasible to run in the real world. To answer (3), we include a comparison to the constrained MDP formulation of Ha et al. [27] by training a critic to predict falls and penalizing the policy for taking actions that the critic predicts will lead to falls. We found that in our case the safety critic required tens of thousands of samples to converge, which was not

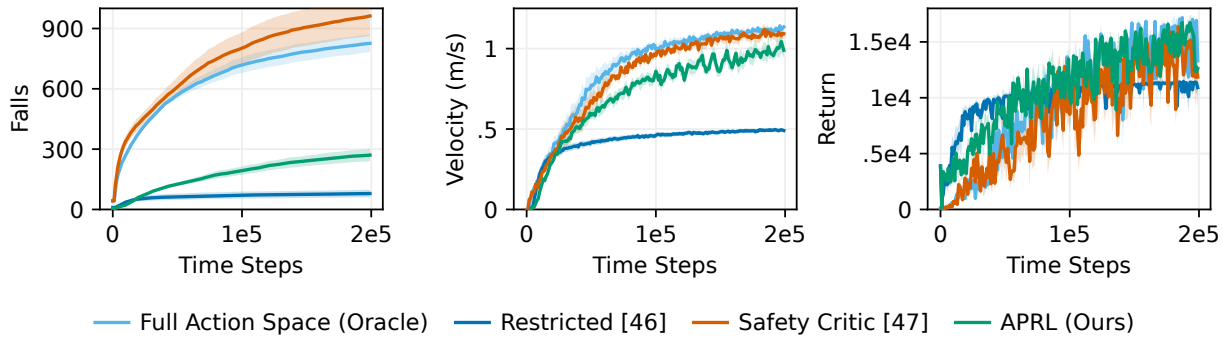


Figure 5.9: **(Simulation comparison results)** We report each policy's performance measured by the falls, average velocity, and return (mean and standard error across 5 seeds) with respect to the number of time steps. We find that APRL is the only method that effectively balances achieving high velocity while regulating the number of falls such that it is feasible to run in the real world.

quick enough to shape the early stage of exploration to prevent excessive falls. In fact, this method was especially sensitive to network initialization and so we omit one seed that diverged for clarity. Generally, it is quite difficult to fit a critic with time-delayed effects whereas our method simply penalizes action magnitudes directly. This method was shown to be effective with a significantly simpler robot, with removed degrees of freedom, where learning a critic may be expected to be much simpler than in our setting.

Finally, we compare APRL's adaptive action regularization to alternatives in order to understand the importance of (a) using a soft constraint rather than a naïve hard constraint (b) using an adaptive rather than fixed expansion rate and (c) regularizing the policy directly rather than through the reward function. For (a), we compare to *Hard Constraint*, which is our method but instead of penalizing the policy outputs, we clip them at the prescribed limits before applying them in the environment. Next, for (b) we test whether the prediction error is meaningfully regulating the speed at which the constraint is changing by only using the constant increment (removing line 11 of Algorithm 5) and label this as *Non-Adaptive Regularization*. Lastly, for (c) we implement a baseline in which we add a very common control cost to the reward function and call this *Reward Regularization*—we follow standard conventions and use a quadratic penalty on the actions. From Fig. 5.9, we see that with non-adaptive regularization there are too many falls which is a major issue for real-world deployment. This problem is exacerbated even further when using a hard constraint. Adding the action penalty via reward regularization causes the policy to exploit the reward getting high return, but with no forward velocity so it does not actually perform the task.

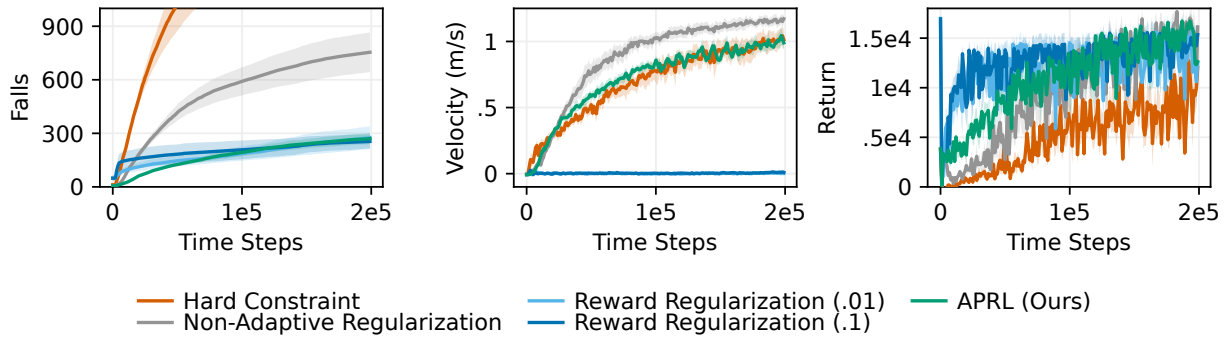


Figure 5.10: **(Simulation ablation results)** We compare to versions of APRL that use (a) a hard constraint (b) non-adaptive regularization (c) regularization via the reward function. These either have too many falls or do not progress on *forward velocity*, showing the importance of all design components of APRL.

5.7 Conclusion

We presented a system for efficiently learning quadrupedal locomotion directly in the real world that improves upon prior work in terms of efficiency and final achieved performance. APRL introduces adaptive policy regularization that encourages the policy to explore within action limits that are commensurate to the policy's competence in a particular situation. APRL allows the robot to effectively utilize its full joint range without causing excessive falling during training. The final speed attained by our policies improves significantly over prior work, both when training from scratch and when fine-tuning to a new terrain.

Our method has several limitations. Although our regularization technique reduces the number of falls, it does not provide a proper "safety" mechanism in the sense that it does not aim to prevent all failures. While this is not a major issue for the small quadrupedal robot we use, it might be a more severe challenge for larger robots. Our method also does not utilize any visual perception, and incorporating this might present additional challenges for sample complexity. Lastly, although the final speed and gait quality acquired by our method improves significantly over the prior approach that learns directly in the real world, the quality of the gaits is still lower than those that have been demonstrated in simulation. Addressing these limitations is an important direction for future work, but we hope that our demonstrated results already indicate that our method represents an important step toward robotic systems that can continually adapt in the real world at deployment time, such that we do not need to train policies that never fail, but can instead allow them to learn to avoid mistakes after they happen.

Part III

Leveraging Priors

Chapter 6

Efficient Online RL with Offline Data

6.1 Motivation

Deep reinforcement learning (RL) has achieved success in a number of complex domains, such as Atari [21] and Go [22], as well as real-world applications like Chip Design [204] and Human Preference Alignment [205]. In many of these settings, strong RL performance is predicated on having large amounts of online interaction with an environment, which is usually made feasible through the use of simulators. In real-world problems, however, we are often confronted with scenarios where samples are expensive, and furthermore, rewards are sparse, often exacerbated by high dimensional state and action spaces.

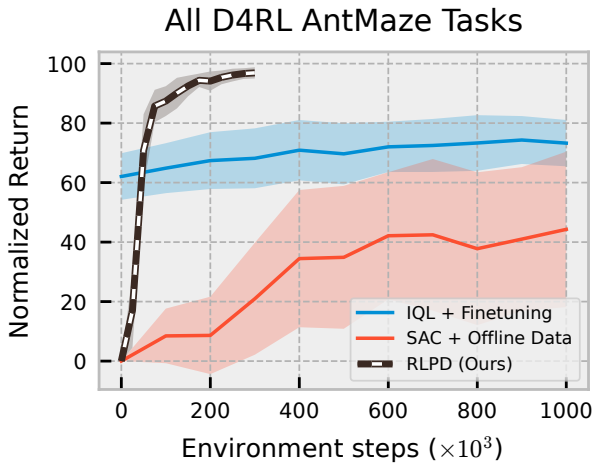


Figure 6.1: **(Overview of RLPD results)** Our approach, RLPD, extends standard off-policy RL and achieves reliable state-of-the-art online performance on several tasks using offline data. Here we show the difficult D4RL AntMaze domain (10 seeds, 1 std. shaded), averaged over all 6 tasks.

One promising way to resolve this issue is via the inclusion of data generated by a previous policy or human expert when training deep RL algorithms (often referred to as *offline data* [206]), as evidenced theoretically [207, 208] and in real-world examples by Cabi et al. [209], Nair et al. [210], Lu et al. [211]. This can alleviate challenges due to sample efficiency and exploration by providing the algorithm with an initial dataset to "kick-start" the learning process, either in the form of high-quality expert demonstrations, or even low-quality but high-coverage exploratory trajectories. This also provides

us with an avenue to leverage large pre-collected datasets in order to learn useful policies.

Some prior work has focused on using this data through pre-training, while other approaches introduce constraints when training online to handle issues with distribution shift. However, each approach has its drawbacks, such as additional training time and hyperparamters, or limited improvement beyond the behavior policy respectively. Taking a step back, we note that *standard off-policy algorithms* should be able to take advantage of this offline data, and furthermore issues with distribution shift should be alleviated in this setting, as we can explore the environment online. Thus far however, such methods have seen limited success in this problem setting. Therefore, we ask the following question: *can we simply apply existing off-policy methods to leverage offline data when learning online, without offline RL pre-training or explicit imitation terms that privilege the prior offline data?*

Through a set of thorough experiments on a collection of widely studied benchmarks, we show that the answer to this question is yes. However, naïvely applying existing online off-policy RL algorithms can result in comparatively poor performance, as we see in Figure 6.1 comparing 'SAC + Offline Data' with 'IQL + Finetuning'. Instead, a minimal set of key *design choices* must be taken into consideration to ensure their success. Concretely, we first introduce a remarkably simple approach to sampling the offline data, which we call "symmetric sampling", that performs well over a large variety of domains with no hyperparameter tuning. Then, we see that in complex settings (e.g., sparse reward, low volume of offline data, high dimensionality, etc.), it is vital that value functions are prevented from over-extrapolation. To this end, we provide a novel perspective on how Layer Normalization [176] implicitly prevents catastrophic value over-extrapolation, thereby greatly improving sample-efficiency and stability in many scenarios, while being a minimal modification to existing approaches. Then, to improve the rate at which the offline data are utilized, we incorporate and compare the latest advances in sample-efficient model-free RL, and find that large ensembles are remarkably effective across a variety of domains. Finally, we identify and provide evidence that key design choices in recent RL literature are in fact environment sensitive, showing the surprising result that environments which share similar properties in fact require entirely different choices, and recommend a *workflow for practitioners* to accelerate their application of our insights to new domains.

We demonstrate that our final approach, which we call **RLPD** (**R**einforcement **L**earning with **P**rior **D**ata) outperforms previously reported results, as we see in Figure 6.1, on many competitive domains, sometimes by $\mathbf{2.5}\times$. Crucially, as our changes are minimal, we maintain the attractive properties of online algorithms, such as ease of implementation and computational efficiency. Furthermore, we see the generality of our approach, which achieves strong performance across a number of diverse offline datasets, from those containing limited expert demonstrations, through to data comprised of high-volume sub-

optimal trajectories.

We believe that our insights are valuable to the community and practitioners. We show that online off-policy RL algorithms can be remarkably effective at learning with offline data. However, we show their reliable performance is predicated on several key design choices, namely the way the offline data are sampled, a crucial way of normalizing the critic update, and using large ensembles to improve sample efficiency. While the individual ingredients of RLPD are refreshingly simple modifications on existing RL components, we show that their combination delivers state-of-the-art performance on a number of popular online RL with offline data benchmarks, exceeds the performance of significantly more complex prior methods, and generalizes to a number of different types of offline data, whether it be expert demonstrations or sub-optimal trajectories. We have released RLPD here: `github.com/ikostrikov/rlpd`.

6.2 Related work

Offline RL pre-training. We note connections to offline RL [206, 212, 213]; many prior works perform offline RL, followed by online fine-tuning [82, 210, 214–216]. Notably, Lee et al. [215] also considers large ensembles and multiple gradient-step per timestep regimes when learning online. However, our approach uses a significantly simpler sampling mechanism with no hyperparameters and does not rely on costly offline pre-training, which introduces yet *additional* hyperparameters. We also emphasize that our normalized update is *not an offline RL method*—we do not perform any offline pre-training but run online RL from scratch with offline data included in a replay buffer.

Constraining to prior data. An alternative to the offline RL pre-training paradigm is to explicitly constrain the online agent updates such that it exhibits behavior that resembles the offline data [214, 217–221]. Particularly relevant to our approach is work by Rajeswaran et al. [220], which augments a policy gradient update with a weighted update that explicitly includes *demonstration* data. In contrast, we use a sample-efficient off-policy paradigm, and *do not* perform any pre-training. Also similar to our work is that by Nair et al. [219], who also use an off-policy algorithm with a fixed offline replay buffer. However, we do not restrict the policy using a behavior cloning term, and do not reset to demonstration states. Moreover, we note that these approaches generally require the offline data to be high quality (i.e., 'learning from demonstration data' [90, 222]), while our approach is, importantly, agnostic to the quality of the data.

Unconstrained methods with prior data. Prior work has also considered ways of incorporating offline data without any constraints. Some methods focus on initializing a replay buffer with offline data [214, 223], while other works have utilized a balanced

sampling strategy to handle online and offline data [82, 219, 224, 225]. Most recently, Song et al. [208] presented a theoretical analysis of such approaches, showing that balanced sampling is important both in theory and practice. In our experiments, we also show that balanced sampling helps online RL with offline data; however, directly using this approach on a range of benchmark tasks is insufficient, and other design decisions that we present are critical in achieving good performance across all tasks.

6.3 Preliminaries

We consider problems that can be formulated as a Markov Decision Process (MDP) [226], described as a tuple $(\mathcal{S}, \mathcal{A}, \gamma, p, r, d_0)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space and $\gamma \in (0, 1)$ is the discount factor. The dynamics are governed by a transition function $p(s'|s, a)$; there is a reward function $r(s, a)$ and initial state distribution $d_0(s)$. The goal of RL is then to maximize the expected sum of discounted rewards: $\mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)\right]$.

In this chapter, we focus on RL while having access to offline datasets $\mathcal{D}$ [206], a collection of (s, a, r, s') tuples generated from a particular MDP. A key property of offline datasets is they usually do not provide complete state-action coverage, i.e., $\{s, a \in \mathcal{D}\}$ is a small subset of $\mathcal{S} \times \mathcal{A}$. Due to this lack of on-policy coverage, methods using function approximation may over-extrapolate values when learning on this data, leading to a pronounced effect on learning performance [213].

6.4 Online RL with Offline Data

As outlined in Section 6.3, we consider the standard RL setting with the addition of a pre-collected dataset. In this chapter, we aim to design a general approach that is agnostic to the *quality and quantity* of this pre-collected data. For instance, this data could take the form of a handful of human demonstrations, or swathes of sub-optimal, exploratory data. Furthermore, we wish to make recommendations that are agnostic to the nature of the problem setting, such as whether the observations are state or pixel-based, or whether the rewards are sparse or dense.

To this end, we present an approach based on off-policy model-free RL, without pre-training or explicit constraints, which we call **RLPD** (**R**einforcement **L**earning with **P**rior **D**ata). As discussed in Subsection 6.4.5, we base our algorithm design on SAC [72, 227], though in principle these design choices may improve other off-policy RL approaches. First, we propose a simple mechanism for incorporating the prior data. Then, we identify a pathology that exists when naïvely applying off-policy methods to this problem setting, and propose a simple and minimally invasive solution. After, we improve the rate the offline data are utilized by incorporating the latest approaches in sample-efficient RL.

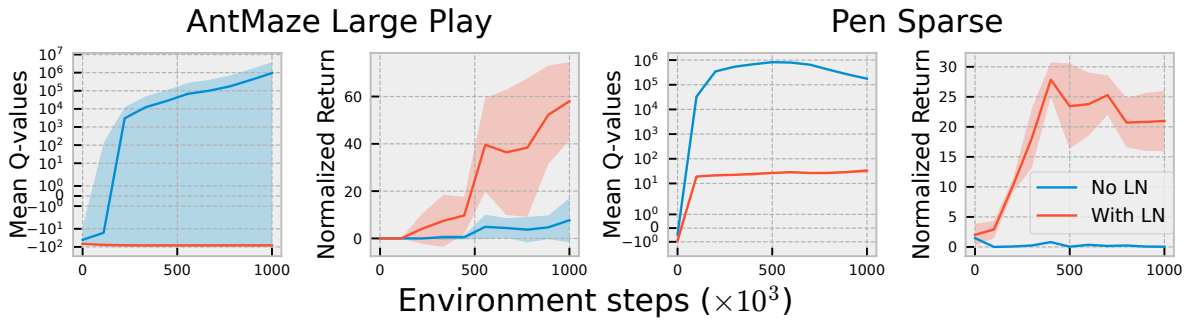


Figure 6.2: **(LayerNorm Ablation)** Using SAC with our symmetric sampling method can result in instabilities due to diverging Q-values; with LayerNorm in the critic this disappears, improving performance.

Finally, we highlight common design choices in recent deep RL that are in fact environment sensitive, and should be adjusted accordingly by practitioners.

6.4.1 Design Choice 1: A Simple and Efficient Strategy to Incorporate Offline Data

We start with a simple approach that incorporates prior data which adds no computational overhead, yet is agnostic to the nature of the offline data. We call this 'symmetric sampling', whereby for each batch we sample 50% of the data from our replay buffer, and the *remaining 50% from the offline data buffer*, resembling the scheme used by Ross and Bagnell [228]. As we will see in later sections, this sampling strategy is surprisingly effective across a variety of scenarios, and we extensively ablate various elements of this scheme (see Subsection 6.5.1). However, applying this approach to canonical off-policy methods, such as SAC [72], does not yield strong performance, as we see in Figure 6.1, and further design choices must be taken into consideration.

6.4.2 Design Choice 2: Layer Normalization Mitigates Catastrophic Overestimation

Standard off-policy RL algorithms query the learned Q-function for out-of-distribution (OOD) actions, which might not be defined during learning. Consequently, there can be significant overestimation of actual values due to the use of function approximation [229]. In practice, this phenomenon leads to training instabilities and possible *divergence* when the critic is trying the catch up with a constantly increasing value.

In particular, we find this to be the case when naïvely applying our symmetric sampling approach for complex tasks (see Figure 6.2). Critic divergence is a well-studied problem,

particularly in the *offline* regime, where the policy cannot generate new experience. In our problem setting, however, we *can* sample from the environment. Therefore, instead of creating a mechanism that explicitly discourages OOD actions, which can be viewed as anti-exploration [230], we instead need to simply ensure that the learned functions do not extrapolate in an unconstrained manner. To this end, we show that Layer Normalization (LayerNorm) [176] can bound the extrapolation of networks but, crucially, *does not* explicitly constrain the policy to remain close to the offline data. This in turn does not discourage the policy from exploring unknown and *potentially valuable* regions of the state-action space. In particular, we demonstrate that LayerNorm bounds the values and empirically prevents catastrophic value extrapolation. Concretely, consider a Q-function Q parameterized by θ, w, applying LayerNorm and intermediate representation $\psi_\theta(\cdot, \cdot)$. For any a and s we can say[1]:

$$\begin{aligned}\|Q_{\theta,w}(s,a)\| &= \|w^T \text{relu}(\psi_\theta(s,a))\| \\ &\leq \|w\| \|\text{relu}(\psi_\theta(s,a))\| \leq \|w\| \|\psi(s,a)\| \\ &\leq \|w\|\end{aligned}$$

Therefore, as a result of *Layer Normalization*, the Q-values are bounded by the norm of the weight layer, even for actions outside the dataset. Thus, the effect of erroneous action extrapolation is greatly mitigated, as their Q-values are unlikely to be significantly greater than those already seen in the data. Indeed, referring back to Figure 6.2, we see that introducing LayerNorm into the critic greatly improves performance through mitigating critic divergence.

To illustrate this, we generate a dataset with inputs x distributed in a circle with radius 0.5 and labels $y = \|x\|$. We study how a standard two-layer MLP with ReLU activations (common in deep RL) extrapolates outside of the data distribution, and the effect of adding LayerNorm. In Figure 6.3, the standard parameterization leads to unbounded extrapolation outside of the support, while LayerNorm bounds the values, greatly reducing the effect of uncontrolled extrapolation.

6.4.3 Design Choice 3: Sample Efficient RL

We now have an online approach leveraging offline data that also suppresses extreme value extrapolation, whilst maintaining the freedom of an unconstrained off-policy method. However, a benefit of offline and constrained approaches is that they have an explicit mechanism to efficiently incorporate prior data, such as through pre-training [214, 215], or an auxiliary supervision term [219, 221] respectively. In our case, the incorporation of prior data is implicit through the use of online Bellman backups over offline transitions.

[1]For simplicity, we consider LayerNorm without bias terms. This does not change the analysis, as it is a constant.

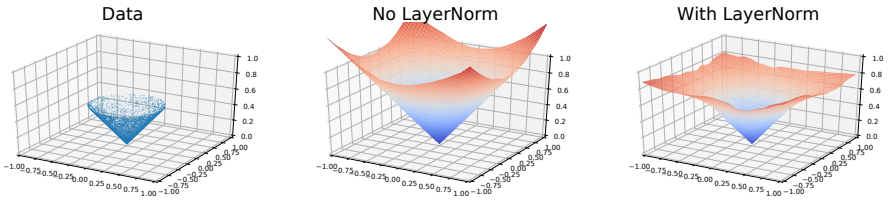


Figure 6.3: **(Didactic example of extrapolation with and without LayerNorm)** We fit data (left) with a two-layer MLP without LayerNorm (center) and with LayerNorm (right). LayerNorm bounds the values and prevents catastrophic overestimation.

Therefore, it is imperative that these Bellman backups are performed as sample-efficiently as possible.

One way to achieve this is to increase the number of updates we perform per environment step (also referred to as update-to-data (UTD) ratio), allowing the offline data to become "backed-up" more quickly. However, as highlighted in recent literature in online RL, this can create issues in the optimization process and ironically *reduce sample efficiency*, due to statistical over-fitting [202]. To ameliorate this, prior work has suggested a number of regularization approaches, such as simple L2 normalization [223], Dropout [174, 231] and random ensemble distillation [58]. In this chapter, we settle on the latter approach of random ensemble distillation; we will demonstrate through ablations that this performs strongest, particularly on sparse reward tasks.

We also note that value over-fitting issues exist when performing TD-learning from images [232]. Therefore in these settings, we further include random shift augmentations [233, 234].

6.4.4 Per-Environment Design Choices

Having highlighted the 3 key design choices for our approach that can be applied generally to all environments and offline datasets, we now to turn our attention to design choices that are commonly taken for granted, but can in fact be environment-sensitive. It is well documented that deep RL algorithms are sensitive to implementation details [235–238]. As a result, many works in deep RL require per-environment hyperparameter tuning. Given the huge variety of tasks we consider in our experiments, we believe it is important to contribute to this discourse, and highlight that certain design choices, which are often simply inherited from previous implementations, should in fact be *carefully reconsidered*, and may explain why off-policy methods have not been competitive thus far on the problems we consider. We therefore take a view that, given the well-documented sensitivity of deep RL, it is important to demonstrate a critical path of design choices

to consider when assessing new environments, and provide a *workflow* to simplify this process for practitioners.

Clipped Double Q-Learning (CDQ). Value-based methods combined with function approximation and stochastic optimization suffer from estimation uncertainty, which, in combination with the maximization objective of Q-learning, leads to value overestimation [239] (as also discussed in Subsection 6.4.2). In order to mitigate this issue, Fujimoto et al. [240] introduce Clipped Double Q-Learning (CDQ) which involves taking a minimum of an ensemble of two Q-functions for computing TD-backups. They define the targets for updating the critics as follows:

$$y = r(s, a) + \gamma \min_{i=1,2} Q_{\theta_i}(s', a') \text{ where } a' \sim \pi(\cdot|s').$$

However, this corresponds to fitting target Q-values that are 1 std. below the actual target values, and recent work [241] suggests that this design choice may not be universally useful as it can be too conservative. Therefore, this is important to reconsider, especially outside of the domains for which it was originally designed, such as sparse reward tasks prevalent in our problem setting.

Maximum Entropy RL. MaxEnt RL augments the traditional return objective with an entropy term:

$$\max_{\pi} \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t (r_t + \alpha \mathcal{H}(\pi(a|s))) \right].$$

This corresponds to maximizing the discounted reward and expected policy entropy at each time step. The motivation for these approaches is centered around robustness and exploration (i.e., maximize reward while behaving as randomly as possible). Approaches relying on this objective are empirically impressive [58, 72, 174, 227]. We, therefore, believe this design choice is of interest in the context of online fine-tuning, where rewards are often sparse and require exploration.

Architecture. Network architecture can have a significant impact on deep RL performance, and the same architecture that can be optimal in one environment can be sub-optimal in another [238]. To simplify the search space, we consider the impact of having 2 or 3 layers in the actor and critic, which have been shown to affect performance, even on canonical tasks [242].

6.4.5 RLPD: Approach Overview

Here we present pseudo-code for our approach, highlighting in Green elements that are important to our approach, and in Purple, environment-specific design choices.

Algorithm 6 Online RL with Offline Data (RLPD)

1: Select LayerNorm, ensemble size E, gradient steps G and architecture
2: Initialize critics $\{\theta_i\}_{i=1}^{E}$ and targets $\theta'_i = \theta_i$; initialize actor ϕ; set γ, α, ρ
3: Select critic target subset size $Z \in \{1, 2\}$
4: Initialize replay buffer $\mathcal{R}$
5: Initialize offline data buffer $\mathcal{D}$
6: **while** training **do**
7: Observe s_0
8: **for** $t = 0$ to T **do**
9: Sample action $a_t \sim \pi_\phi(\cdot \mid s_t)$
10: Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{R}$
11: **for** $g = 1$ to G **do**
12: Sample b_R of size $N/2$ from $\mathcal{R}$
13: Sample b_D of size $N/2$ from $\mathcal{D}$
14: Form batch $b = b_R \cup b_D$
15: Sample indices $\mathcal{Z} \subset \{1, \ldots, E\}$ with $|\mathcal{Z}| = Z$
16: Compute targets:

$$y = r + \gamma \min_{i \in \mathcal{Z}} Q_{\theta'_i}(s', \tilde{a}'), \quad \tilde{a}' \sim \pi_\phi(\cdot \mid s')$$

17: Add entropy: $y \leftarrow y + \gamma\alpha \log \pi_\phi(\tilde{a}' \mid s')$
18: **for** $i = 1$ to E **do**
19: Update θ_i by minimizing:

$$L = \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$$

20: Update targets: $\theta'_i \leftarrow \rho\theta'_i + (1 - \rho)\theta_i$
21: Update actor ϕ by maximizing:

$$\frac{1}{E} \sum_{i=1}^{E} Q_{\theta_i}(s, \tilde{a}) - \alpha \log \pi_\phi(\tilde{a} \mid s), \quad \tilde{a} \sim \pi_\phi(\cdot \mid s)$$

The key factors of RLPD reside in lines 1 and 13 of Algorithm 6 with adopting LayerNorm, large ensembles, sample efficient learning and a symmetric sampling approach to incorporate online and offline data. For environment specific choices, we recommend the following as a starting point:

- *Line 3:* Subset 2 critics
- *Line 16:* Remove entropy
- *Line 1:* Utilize a *deeper* 3 layer MLP

As a pragmatic workflow, we recommend ablating these Purple design choices first, and in the order stated above.

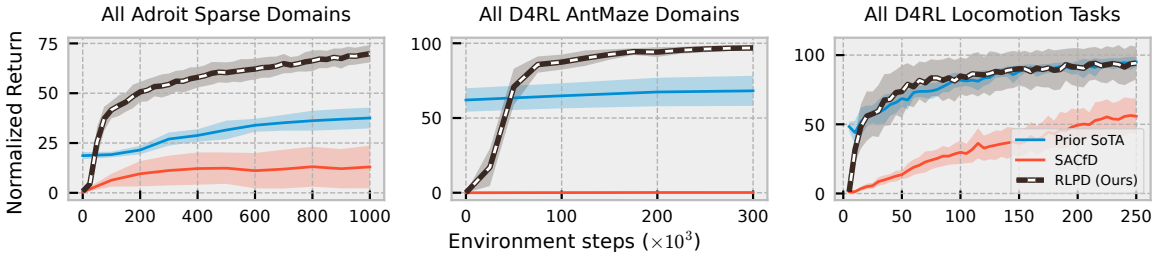


Figure 6.4: **(Results per domain)** RLPD exceeds prior state-of-the-art performance on a number of different popular benchmarks whilst being significantly simpler. Results are aggregated over **21 different environments** (10 Seeds, 1 std. shaded). In each case, we compare to the prior best known work (IQL + Finetuning in Adroit and AntMaze, Off2On in Locomotion), and SACfD, a canonical off-policy approach using offline data.

6.5 Experiments

We design our experiments to not only demonstrate the importance of our design choices, but also provide the insights that allow practitioners to quickly adapt RLPD to their problems. As such, we aim to answer the following questions:

1. Is RLPD competitive with prior work despite using *no pre-training nor having explicit constraints*?
2. Does RLPD transfer to *pixel-based* environments?
3. Does LayerNorm *mitigate value divergence*?
4. Does the proposed workflow around environment-specific design choices lead to *reliable performance*?

For (1), we compare RLPD to those which have been designed to use offline data to accelerate online learning. For (2), we consider an additional suite of tasks to study RLPD's applicability to vision-based domains. Then, for (3) we perform analysis to demonstrate the importance of using LayerNorm. Lastly, for (4) we demonstrate our proposed workflow on the most challenging tasks (see Subsection 6.4.5).

How does RLPD compare? We consider these **21 tasks** from established benchmarks:

- **Sparse Adroit** [210]. These 3 dexterous manipulation tasks—pen-spinning, door-opening, ball relocation—are challenging, sparse-reward tasks. The offline data are multi-modal, with a small set of human demonstrations and a large set of trajectories from a behavior-cloned policy trained on this human data. We follow the *rigorous*

evaluation criteria of Kostrikov et al. [216], whereby performance is based on completion speed, rather than success rate. IQL + Finetuning represents the strongest prior work [216].

- **D4RL AntMaze** [243]. These 6 sparse reward tasks require an Ant agent to learn to walk and navigate through a maze to reach a goal. The offline data comprise *only sub-optimal* trajectories that can in principle be "stitched" together. Again, IQL + Fine-tuning represents the strongest prior work [216].
- **D4RL Locomotion** [243]. Lastly, we have 12 dense reward, locomotion tasks featuring offline data with varying levels of expertise. Off2On [215] has state-of-the-art performance on this suite of tasks.

For evaluation, we first include **SACfD**, a baseline studied in prior work [210, 223], which, similar to RLPD, is an off-policy approach that incorporates offline data during training. However, SACfD simply *initializes* the online replay buffer with the offline data. We implement this baseline using SAC without the additional design decisions discussed in Subsection 6.4.5. Then, since there is no *single* prior method that achieves the best performance across all groups of tasks, we compare to the state-of-the-art method specific to each group (as listed above).We refer to this comparison in our plots as **Prior SoTA**. For all experiments in this chapter, we report the mean and standard deviation across *10 seeds* and aggregate the results across tasks within the three groups listed. For full detailed results broken down by task, see Section 10.5, and for more environment details, see Subsection 10.6.1.

We see that RLPD performs strongly, either matching or *significantly exceeding* the best-known prior work on these challenging benchmarks(see Figure 6.4). We reiterate that we present results for RLPD and SACfD without doing *any pre-training*, unlike the Prior SoTA methods. So, while prior work (shown in blue) may achieve strong initial performance, the online improvement is more modest. On the other hand, our method reaches or surpasses this performance in the order of just 10k online samples. Notably, we outperform the best reported performance on the Sparse Adroit 'Door' task by $\mathbf{2.5}\times$. Moreover, to our knowledge, our method is the first to effectively 'solve' *all* AntMaze tasks. Furthermore, we are able to do so in *less than a third* the time-step budget allocated to prior methods.

Does RLPD transfer to pixels? Here we consider the medium and expert locomotion tasks in V-D4RL [244], an offline dataset with *only pixel observations*. V-D4RL is particularly challenging as the behavior policies are state-based, which means the data is partially observable in pixel-space. This is most obvious in 'Humanoid Walk', where body parts are visually occluded, thus behavior cloning (BC) can struggle to achieve strong performance. For evaluation, as our method seeks to accelerate online learning with offline data, we focus on data-efficiency—we introduce a challenge that we call "10%DMC",

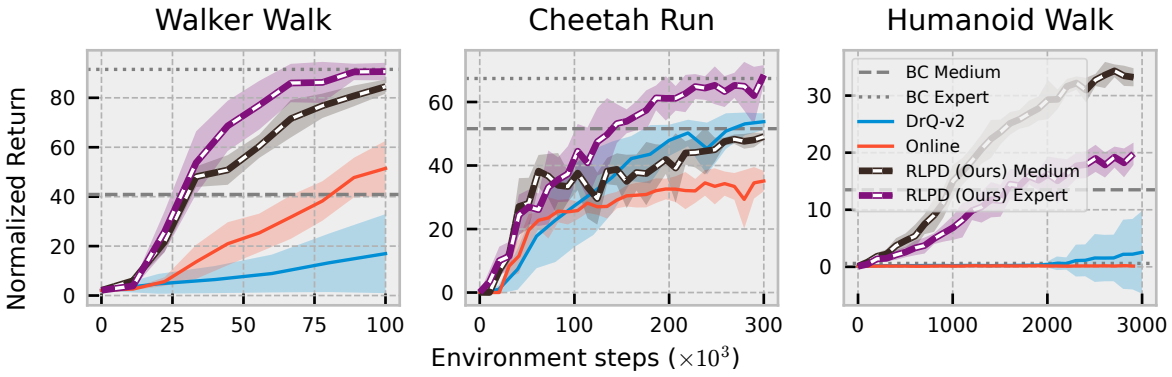


Figure 6.6: **(Results on vision-based tasks)** Our approach generalizes to vision-based domains, providing consistent improvements over existing approaches.

which involves training policies using only 10% of the total timesteps recommended by Yarats et al. [234]. As we use the medium and expert data, we include a **BC** baseline. To evaluate RLPD's ability to efficiently use offline data to boost online learning, we compare to a baseline approach that does not use the offline data. To isolate the effect of the utilization of offline data, we use the same architecture and policy optimizer as our method and label this baseline as **Online** in our plots. Then, to evaluate how this compares to the state-of-the-art sample-efficient RL method from pixels, we compare to **DrQ-v2** [234].

We see in Figure 6.6[2] that RLPD provides consistent improvements over purely online approaches, and in many cases greatly improves over a BC baseline. We see through the difference in RLPD (dashed black and purple lines) and the Online baseline that RLPD *effectively* utilizes the offline data to bootstrap learning. This conclusion holds as we compare to the SoTA vision-based RL method (blue).

Lastly, we test increasing UTD to 10 on one of the tasks—Cheetah Run with Expert offline data. Figure 6.5 shows a remarkable improvement in performance when learning with the offline dataset. To our knowledge, this is the first demonstration of a high UTD approach improving model-free pixel-based continuous control.

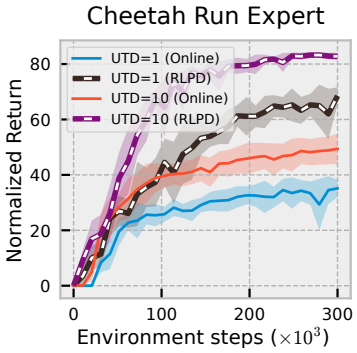


Figure 6.5: **(Effects of changing UTD with vision-based task)** Increasing UTD with RLPD greatly improves sample efficiency from pixels.

6.5.1 RLPD Analysis and Ablation Study

Here, we address (3) and (4) by quantifying the effect of LayerNorm, and demonstrating the reliability of our proposed

[2]V-D4RL normalized return is episode return divided by 10.

workflow (see Subsection 6.4.5).

Does LayerNorm mitigate value divergence? We now illustrate the importance of LayerNorm in mitigating catastrophic value divergence, and focus on tasks where overestimation is particularly prone. In Figure 6.7, we see LayerNorm is crucial for strong performance in the Adroit domain; excluding LayerNorm results in significantly higher variance across seeds and reduces mean performance.

To more clearly illustrate this effect, we construct a dataset of *only the expert human demonstration data* from the Adroit Sparse tasks (see "Expert Adroit Sparse Tasks" in Figure 6.7). This subset comprises just **22** of the 500 trajectories in the original dataset and is much more narrowly distributed by nature—representing a task with sparse rewards, limited demonstrations, and narrow offline data coverage—likely to exacerbate value divergence. Here we see a remarkable result: RLPD still *exceeds* prior work, despite significant restrictions in data. Moreover, removing LayerNorm now results in collapsed performance, with no progress made on any task. We further observe improvements in sample efficiency through the inclusion of LayerNorm in AntMaze and Humanoid Walk through reducing excessive extrapolation. Additional experiments and results are in Section 10.5.

Design choice workflow. We now motivate our workflow in Section 6.4.5, demonstrating the importance of these design choices. We focus on the hardest tasks, namely 'Relocate' in Adroit Sparse, 'Large Diverse' in AntMaze, and 'Humanoid Expert' in V-D4RL, as we found these to be most sensitive. We provide full results on all tasks in Section 10.5, noting that optimal decisions in the harder domains also produce optimal results in the easier domains.

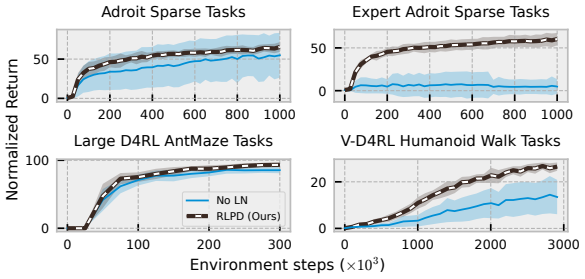


Figure 6.7: **(LayerNorm ablation)** LayerNorm is crucial for strong performance, particularly when data are limited or narrowly distributed.

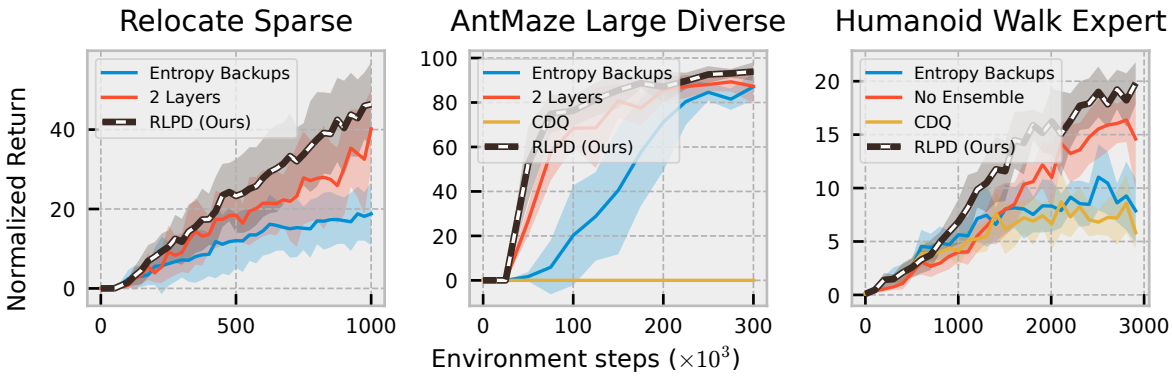


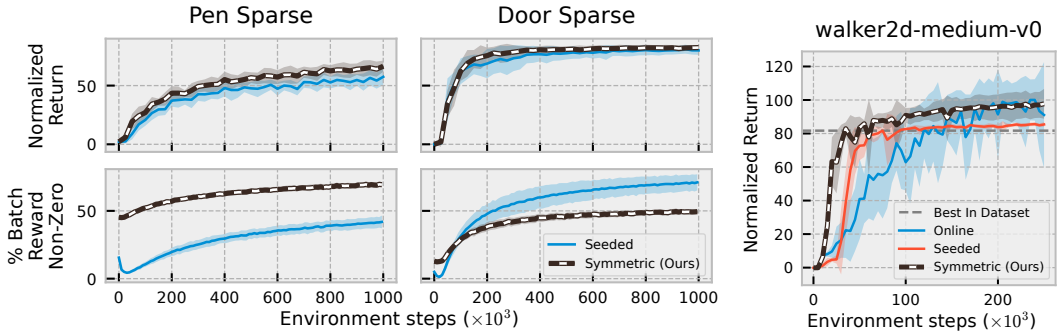
Figure 6.8: **(Empirical evidence for proposed workflow on challenging tasks)** Our recommended starting design choices and workflow leads to strong performance on all tasks.

We see in Figure 6.8 that with the recommended environment-specific design choices provide strong performance; we see using entropy backups and smaller networks always results in worse performance. In 'AntMaze Large Diverse' however we see that with CDQ, performance deteriorates. Following our workflow, and ablating this by subsetting 1 critic is crucial to recovering strong performance. The same applies to 'Humanoid Walk Expert', whereby we surprisingly see that CDQ is detrimental to performance, despite its popularity in recent implementations. We also show the surprising positive effect of larger ensembles in pixel-based tasks, with the standard 2 member critic ensemble performing worse than the 10 member ensemble we use by default in RLPD.

Lastly, we conduct additional ablations to understand the importance of the design choices that we propose for our method, showing that, though the individual design decisions are simple, they are vital for good performance.

Critic regularization. Here we examine the effects of critic regularization on performance. We compare 3 approaches: weight-decay[3] [223], Dropout [174] and ensembling [58]. We choose a subset of the prior experiments to evaluate on from the AntMaze, Adroit, and Locomotion sections.

[3]We found a weight decay value of 0.01 worked best across a number of settings.



(a) Symmetric sampling improves sample efficiency and reduces variance across seeds without merely increasing reward density in the batch.

(b) Initializing the buffer with large amounts of data limits improvement.

Figure 6.10: **(Buffer sampling analysis)** Comparing the effects of design decisions in sampling from the offline data.

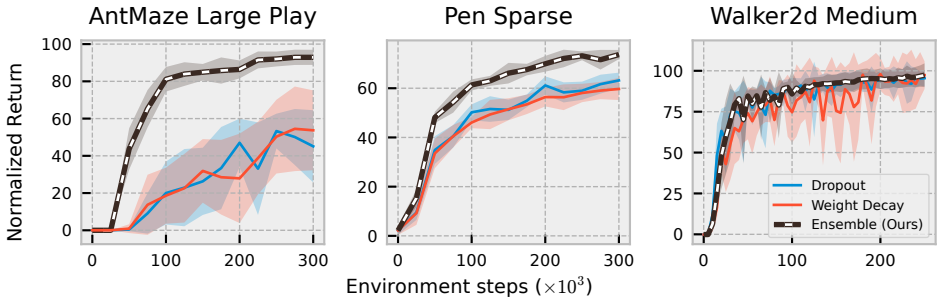


Figure 6.9: **(Critic regularization analysis)** In general, critic ensembling provides the best performance. Dropout performs worse in sparse reward tasks.

In Figure 6.9, we see that ensembling is the strongest form of regularization. Notably, while Dropout performs well in the Locomotion domain 'walker2d-medium-v0', as affirmed by Hiraoka et al. [174], it does not generalize to challenging sparse reward environments. We also see that weight-decay regularization is less performant in all domains.

Buffer initialization. In this section, we compare symmetric sampling with that of one which relies on *initializing* a replay buffer with offline data. First returning to the challenging Human Expert Demonstrations setting in Adroit, in Figure 6.10a we show two contrasting examples that demonstrate symmetric sampling effectively trades-off between replay and offline data. We observe that in the Pen environment, symmetric sampling clearly improves exploration by ensuring increasing reward density within mini-batches. In contrast, we see that although the buffer initialization approach explores just as well in the Door environment, symmetric sampling has the important effect of improving stability

and decreases variance due to relying less on the higher-variance data generated by the online policy.

We now consider a situation whereby we have abundant sub-optimal offline data, and see that again, our balanced sampling approach improves sample-efficiency. In Figure 6.10b, initializing the buffer with high volumes of medium quality locomotion data gives initial improvement over online performance, but struggles asymptotically, likely due to a lack of on-policy data sampling, vital for online improvement. On the other hand, our symmetric sampling approach improves sample efficiency and matches asymptotic performance while reducing variance.

Sampling proportion sensitivity. We assess the sensitivity of our sampling approach from the "symmetric" ratio of 50% online/offline.

As we see in Figure 6.11, RLPD is not very sensitive to sampling proportion. While sampling 25% offline can marginally help asymptotic performance in 'walker2d-medium-v0', this comes at the expense of variance and sample efficiency in sparse reward tasks. We also affirm with our 100% offline results that RLPD is not an offline method, and key to its success is how it controls for divergence *without restricting exploration or behavior learning.*

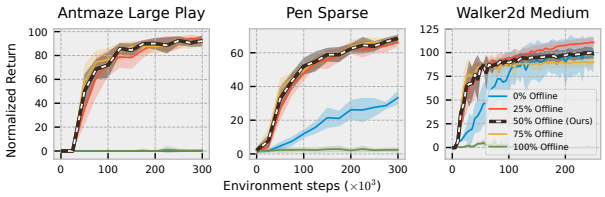


Figure 6.11: **(Effect of replay proportion)** RLPD is not sensitive to replay proportion; 50% offers the best compromise between variance, speed of convergence, and asymptotic performance.

6.6 Conclusion

In this chapter, we show that off-policy approaches can be adapted to leverage offline data when training online. We see that with careful application of key design choices, RLPD can attain remarkably strong performance, and demonstrate this on a total of **30 different tasks**. Concretely, we show that the unique combination of symmetric sampling, LayerNorm as a value extrapolation regularizer, and sample efficient learning is key to its success, resulting in our outperforming prior work by up to **2.5**× on a large variety of competitive benchmarks. Moreover, our recommendations have negligible impact on computational efficiency compared to pure-online approaches, and are simple, allowing practitioners to easily incorporate the insights presented in this chapter into existing approaches. To further improve adoptability, we recommend and demonstrate a workflow for practitioners that improves performance on a wide variety of tasks, and thus demonstrate that certain canonical design choices should be reconsidered when applying off-policy methods. Finally, to facilitate future research, we have released the RLPD

codebase here: `github.com/ikostrikov/rlpd`, which features highly optimized off-policy algorithms for proprioceptive and pixel-based tasks in a single codebase written in JAX [178].

Chapter 7

Learning and Adapting Agile Locomotion Skills by Transferring Experience

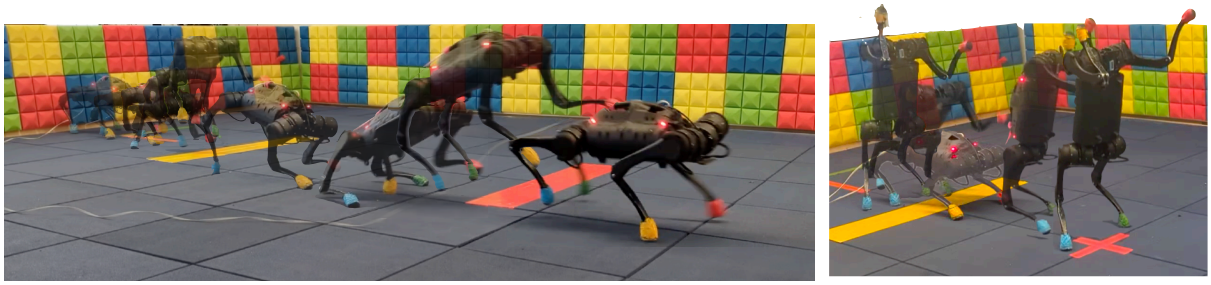


Figure 7.1: **(Agile skills learned with TWiRL)** TWiRL enables the A1 robot to jump repeatedly (left) and walk to a goal location on its hind legs (right).

7.1 Motivation

Legged animals are capable of impressive displays of agility, from mountain goats balancing on sheer inclines to search and rescue dogs bounding through rubble. Emulating such coordination could enable legged robots to go almost anywhere humans can; however, doing so is a long-standing challenge in robotics. It requires the ability to control, at high frequency, robotic systems that are high-dimensional, underactuated, and difficult to model—especially in the case of agile skills with aerial phases and unexpected contacts. Traditional, model-based control methods have demonstrated impressive agility on legged

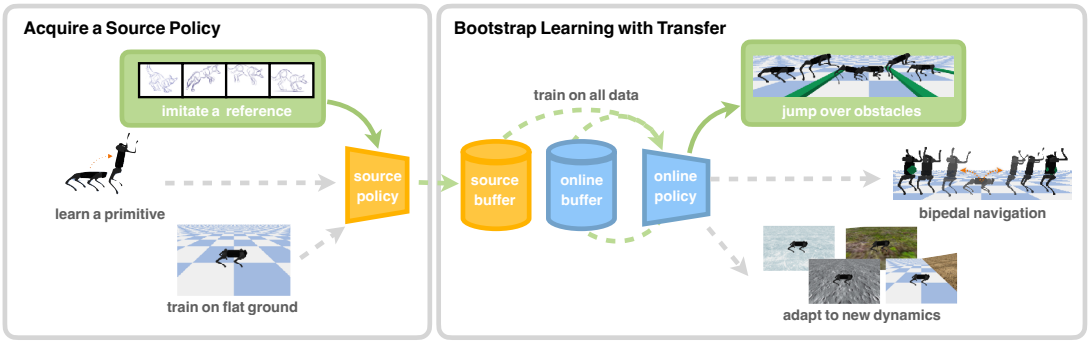


Figure 7.2: **(Overview of practical applications)** We identify that several fundamental challenges in learning agile locomotion skills can be ameliorated by casting them as transfer learning problems, then applying a simple, generic method that involves a simple modification to off-the-shelf off-policy algorithms. We show that our framework is versatile—with the same method, we can **(top)** generalize a policy trained to track a reference motion of a jumping dog to learn to jump over randomly placed obstacles; **(middle)** take a policy that is trained to kick up onto the robot's hind legs to then use bipedal locomotion to navigate to randomly sampled goals; and **(bottom)** enable efficient fine-tuning in new environments.

robots, but they often require extensive knowledge of the robot and environment dynamics, as well as task-specific knowledge about the desired agile movement [186, 187, 245–250].

Reinforcement learning (RL), on the other hand, provides a powerful framework for autonomously acquiring robotic skills. However, learning agile locomotion policies end-to-end remains challenging for a few fundamental reasons. Foremost is that tasks with such high-dimensional systems—12 degrees of freedom for the A1 quadruped—are woefully underspecified, e.g, a robot can accomplish the simple task of moving forward in innumerable ways that are unnatural and dangerous. It is well known that even simple tasks can often require very complex, highly-shaped reward functions to elicit desired motions [32, 57, 77, 181]. This underspecification problem is exacerbated by the additional complexity demanded by skills that require high speed and precision. Consider learning to jump over hurdles—the robot must first discover the coordination to launch itself in order to make any sort of meaningful progress—compared to walking, where fumbling forward still contains some useful information about how to move. That is to say, learning in this regime without extremely dense reward engineering is especially difficult because the robot can only receive learning signal by engaging in already-sophisticated, temporally coherent, and precise behaviors. So learning skills that require precise coordination of joints and contact forces entirely from scratch may be prohibitively difficult. How might we acquire agile skills in light of these challenges?

While it may be difficult to obtain *demonstrations* for a specific task (e.g. jumping

to clear past variable obstructions, squeezing through tight areas, or maneuvering in extreme weather conditions), it is often possible to obtain controllers for adjacent, but simpler, settings—by training with different objectives, sensory inputs, or environments—that nonetheless contain *relevant* information. Providing the robot with such relevant knowledge should significantly reduce the complexity of the learning problem by side-stepping the difficult, and sometimes insurmountable, exploration hurdle. For example, Figure 7.1 shows a task where the robot needs to walk to a specific location on its hind legs. Learning this task should be made significantly easier if the robot can already stand on its hind legs, a comparatively simpler behavior to engineer. But these simpler behaviors are often difficult to then "relax" into more agile ones, especially if they are trained with heavy reward shaping to give rise to their desirable behavior. Thus, to learn these more complex skills, we would like to somehow "pre-train" with the simpler or more well-shaped objectives, and then fine-tune with a minimally shaped objective that encodes the actual task we would like to solve, allowing RL to modify the motion as much as it needs to in order to achieve optimality. Adapting to new environments looks very similar—if we already have an agile skill, we might similarly prefer to fine-tune it rather than learn from scratch. In all cases, we need to answer the question: how should we train for a desired final task given another policy pre-trained on a different task, in a different environment, or with a different objective?

The main contribution described in this chapter is a system for training agile robotic skills, such as jumping and walking on hind legs, facilitated by a transfer learning procedure that enables effective initialization of one skill with existing suboptimal source skills. To be broadly applicable, we present a simple, general framework for initializing skills that aims to (i) take advantage of a wide range of 'source' controllers, treating them as black boxes that can be queried without requiring knowledge of their specific implementations and (ii) contend with the suboptimality of these controllers with respect to the desired task. As further detailed in Section 7.4, this transfer learning procedure leverages off-policy RL methods to effectively learn from the experiences generated by source controllers. We experimentally demonstrate that this provides a broadly applicable, and highly effective way to initialize skills in a target Markov decision process (MDP) with skills from other source MDPs. We then demonstrate that this can be used to provide a curriculum where simpler shaped reward and motion imitation skills can be used to bootstrap learning of highly agile behaviors, such as jumping over multiple obstacles and walking on the hind legs, and that the same exact framework can be used to transfer these skills to different environments. Finally, we show executing the learned policies enables—for the first time—a real A1 quadrupedal robot to perform consecutive running jumps across randomly placed hurdles, crossing each span of 20 cm in a *single* flight phase and to walk only with its hind legs, balancing precariously on at most two of its spherical rubber feet.

7.2 Related Work

Developing control strategies for legged robots has been a long-standing research topic in robotics. Model-based methods that leverage trajectory optimization [186, 187] and model-predictive control (MPC) [44, 46, 59, 60, 188, 189] tackle this problem by using manually constructed dynamics models of the robot. These methods have shown compelling results on robust control of multiple gaits [249] and adaptation to complex terrains [251–253]. Yet, modeling the dynamics of the robot usually requires significant expertise and engineering effort for individual problem settings. Learning-based approaches offer an appealing alternative, potentially automating the process of obtaining control policies—without requiring explicit modeling of the dynamics—and have been shown to consistently excel at enabling robust legged locomotion in simulation [53–55] and even in the real world [28, 31, 34, 47, 48, 50, 56, 181, 254, 255].

In this chapter, we tackle *agile* locomotion—specifically running-jump behaviors as well as bipedal navigation with a quadrupedal robot. Recently purely model-based approaches have demonstrated in the real world a variety of highly performant jumping controllers for the MIT Cheetah 2 [256, 257], MiniCheetah [249] and A1 [258] with MPC; however, these rely on multistage planners specialized for the particular jumping motion and assume access to the dynamics model. Some work has used RL, leveraging MPC controllers to jump [193, 259]. As for fully learning-based approaches, Rudin et al. [192] learn a jumping controller focused on reorientation and landing in micro-gravity environments. In terms of comparable results to ours, Margolis et al. [193] demonstrate that using high-level velocity and foot force commands can enable a MiniCheetah to continuously hop over randomly placed gaps. When transferred to the real world, the robot is able to cross 26cm gaps in two hops (the first to center its body over the gap and transfer its front feet across and the second to then transfer its back feet). In contrast, we utilize low-level PD target actions and can directly apply our method to learn other motions (such as bipedal locomotion). We further demonstrate an A1 robot clearing 20cm gaps in the real world in *one* leap, moving its entire body over the gap in one highly dynamic motion. As for bipedalism with quadrupedal robots, Vollenweider et al. [195], who use adversarial motion priors to get an ANYmal to balance on its hind legs with large wheels as feet and do not then demonstrate walking. Yu and Rosendo [194] demonstrate bipedal walking with a quadrupedal robot; however, they add a supporting stick to both back feet, making it possible to get up quasi-statically and still have 4 points of contact while walking, so the policy does not require a high degree of agility. Fuchioka et al. [196] use RL to imitate a reference motion of bipedal stepping generated with trajectory optimization on a Solo 8 robot [260]. As we discuss below, our work is complementary to such imitation-based policies since our approach is to use their data to learn more complex tasks.

As mentioned in Section 7.1, the primary challenges in learning high dimensional un-

deractuated robotic systems are task specification and exploration. A popular way to address these two challenges is imitation learning [90–93], where the agent is trained to copy an expert. In locomotion, this principle often manifests as tracking reference motions [57, 196, 261–265]. The reference motion in these cases specifies the task completely and provides very dense feedback, addressing the exploration problem. The drawback of imitation-based approaches, though, is that in exchange for solving the exploration challenge, they are restricted to the "expert" or reference motion as the policy's objective is to track it. That is, they are not *directly* applicable to solve downstream tasks or be adapted to situations in which the expert performs sub-optimally [266, 267]. Bogdanovic et al. [268] make a similar observation that motion-tracking can be brittle to environment uncertainties, and so first use an imitation objective to learn a policy that is then fine-tuned with randomization on an actual task objective. Our work specifically explores initializing the replay buffer as an effective avenue for cross-task information transfer, with the benefit of being able to utilize data from existing policies (with different action and observation spaces) without having to first train with a motion-tracking objective.

When reference motions are unavailable, another option is heavy reward engineering, with or without a curriculum. Rudin et al. [33] report using a 9-term reward function to learn a velocity-conditioned locomotion task, with manually-tuned weights to balance the various reward terms. This method can also yield impressive, more agile controllers, with the investment of sufficient effort in tuning the parameters. Margolis et al. [191] use a similar setup with an adaptive curriculum on the velocity command to achieve a 3.9 m/s controller for the MiniCheetah. Several works show promising results using automatic curricula without the need to encode extensive domain knowledge, but they have yet to be demonstrated outside of simulation [269, 270]. These are all valid ways to obtain controllers, and we in fact leverage some of them in our own work. Our method is concerned with bootstrapping *from* existing policies to accomplish more difficult tasks. Therefore, any of these existing training methods can be seen as complementary to our method.

Our method applies the concept of transfer using a general method in order to address the aforementioned challenges. One facet of transfer very commonly studied in the realm of locomotion is that of shifts in dynamics (e.g., sim-to-real, different types of terrains). Domain randomization is widely used as a method for achieving generalizable policies by training them to do well on average under sufficiently diverse conditions [12, 14, 15, 29, 57, 61, 65, 67, 156, 157] so as to ensure that the resulting policy can be adapted to variations of its training environment. Meta-RL is another line of work that achieves few-shot adaptation by directly optimizing the adaptation procedure [158–161]. These approaches all require some sort of distribution of training conditions, and an assumption that the test conditions will be likely under this distribution. In order to be able to leverage existing controllers, we cannot make such assumptions on the training distribution or origins of

our source policies. As such, we use the *experience*, which we can get by running the source policies in our target domain, to bootstrap learning from

While there is a breadth of work in initializing agents with prior data to bootstrap from [90, 157, 219, 221, 222, 271], these methods often make the assumption that the source policy is optimal and *constrain* the agent to behave similarly to the demonstration data. The nature of these problems is that the demonstrations are assumed to be optimal with respect to the task, and the RL is needed in order to learn a policy that is robust to variations. Most similar in spirit to our work is DDPGfD [223] and Nair et al. [219], in that they leverage off-policy RL to use a pre-collected dataset to overcome exploration challenges. They don't, however, consider changes to the MDP like we do to apply this principle to learning agile locomotion. Xie and Finn [272] employ a similar methodology to ours in the continual learning setting while leveraging functional access to the new task's reward to relabel past experiences and filtering to only continue learning on transitions that are similar to the online data. They demonstrate results on a sequence of manipulation tasks with end-effector control at 5Hz, we instead study the ability to apply this principle to very difficult locomotion skills with low-level control at 20Hz.

7.3 Preliminaries

We frame learning a locomotion skill as a Markov decision process (MDP) [226] $\mathcal{M}$, defined as a tuple $(\mathcal{S}, \mathcal{A}, \gamma, p, r)$, describing an agent situated in an environment with state space $\mathcal{S}$ and action space $\mathcal{A}$, whose dynamics are governed by a transition function $p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$. Given a reward function $r(\mathbf{s}, \mathbf{a})$ and discount factor γ, the RL objective is to learn a policy π that maximizes the agent's expected discounted return: $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$.

Transfer learning We are interested in the setting where we have access to a policy π_{src} trained in a source MDP $\mathcal{M}_{\text{src}}$, which is structurally related to the desired target MDP $\mathcal{M}$. For example, $\mathcal{M}_{\text{src}}$ might have a different reward function (e.g., imitating a jumping motion rather than actually clearing an obstacle), a different state space (e.g., $\mathcal{M}$ includes a desired target to walk to, but $\mathcal{M}_{\text{src}}$ does not), or different dynamics (e.g., $\mathcal{M}_{\text{src}}$ involves jumping on flat ground, but $\mathcal{M}$ requires jumping on sloping ground). This includes both curriculum learning problems, where the policy might be difficult to learn in $\mathcal{M}$ directly but easier when appropriately leveraging π_{src}, and problems where we might want to accelerate learning of π by leveraging a π_{src} that is already performant in some other setting. In all of these cases, we would like to use π_{src} to aid in learning π in the target MDP $\mathcal{M}$, but it is not obvious how this should be done. As we will discuss and illustrate in our experiments, naïvely initializing π from π_{src} is not always possible, and often is not the best choice.

Off-policy RL While there are many algorithms for optimizing this objective, we specifically consider *off-policy* methods, as they can in principle incorporate data from other sources—like those collected by hand-designed, simpler, or more narrow policies. Many off-policy deep RL algorithms for continuous control [72, 177] fit a critic using a dataset of experiences to estimate its performance and then update the policy accordingly. This dataset of experiences, or replay buffer $\mathcal{D}$, is usually composed of the robot's own past experiences while learning in a particular MDP. We will discuss how such methods can be used as part of a transfer learning procedure in several different ways, and present a specific approach that we find to work well across a variety of agile quadrupedal control problems.

7.4 Transfer Learning for Agile Skills

We present TWiRL: **T**ransferring **Wi**th off-policy **RL**, a simple yet effective model-free RL framework for learning complex locomotion skills. In this section, we argue that transfer learning provides a very powerful tool for addressing the exploration challenges outlined in Section 7.1 and thus makes it feasible to learn highly agile robotic locomotion skills. We then detail a practical method for facilitating this transfer through simple modifications to off-policy RL.

7.4.1 Bootstrapping Capable and Flexible Policies

Exploration is particularly daunting for agile locomotion due to the complex coordination and precision required combined with the aforementioned challenges in task specification and high-dimensional control. We propose a method to leverage existing controllers to help overcome this challenge; however, it is difficult to devise a method that can adapt policies trained in MDPs that can differ in various ways from the target MDP. First, while they may be useful in providing examples of decent behaviors, out-of-the-box policies are likely highly suboptimal for the target task. As agile skills require precision, the robot should be able to pick out *only* what is useful, adapting or building upon it to accomplish the target task without inheriting idiosyncrasies or irrelevant behaviors. A natural idea is to simply *fine-tune* policies for the new task, but this may not always be possible, especially in the applications we consider. In the case of transferring policies trained with different objectives, the observation space will often change *along with* the reward function as the robot may need additional information to solve a task. For example, we may want to use a blind (trained only from proprioception) policy that has focused on learning a difficult control problem to aid in learning a task that requires knowledge of its surroundings. Another case in which we cannot just straightforwardly fine-tune a policy is if we want to bootstrap learning from policies that were not learned via RL, e.g., traditional model-based methods [187, 188, 249, 251]. To handle all these cases, a key

Algorithm 7 TWiRL pseudocode

Require: Source policy π_{src}, MDP $\mathcal{M}_{\text{src}}$, MDP $\mathcal{M}_{\text{target}}$ of the desired task

Initialize: Source policy replay buffer $\mathcal{D}_{\text{src}}$, online replay buffer $\mathcal{D}_{\text{target}}$, sampling ratio ϕ, policy parameters θ

// Obtain data from π_{SRC}
if $\mathcal{D}_{\text{src}}$ is empty **then**
    repeat
        Collect data with $\pi_{\text{src}}(\mathbf{a} \mid \mathbf{s})$.
        Add experience to source buffer $\mathcal{D}_{\text{src}} \leftarrow (\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
    until desired

// Train on all data using high UTD off-policy RL
repeat
    Collect data with $\pi_\theta(\mathbf{a} \mid \mathbf{s})$.
    Add experience to online buffer $\mathcal{D} \leftarrow (\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
    Construct β_{src} sampling ϕ·batch size from $\mathcal{D}_{\text{src}}$
    Construct β_{target} sampling $(1-\phi)$·batch size from $\mathcal{D}$
    Update θ using $\beta_{\text{src}} \cup \beta_{\text{target}}$
until converged

desideratum of our system is it should be agnostic to *how* the source policy is obtained.

7.4.2 TWiRL: Approach Overview

Our key insight is that transferring *experience* provides a general avenue through which to bias the robot's exploration without constraining it. The only assumption we make is the ability to execute actions from the source policy in $\mathcal{M}$ and record the observations, actions, and rewards. The question now is how we can effectively leverage this possibly suboptimal data to bootstrap learning. Since off-policy RL uses dynamic programming to propagate the information, it can in principle connect its online experience to relevant parts of the source policy data and quickly coopt effective strategies exhibited by the source policy. We find empirically that incorporating the data from π_{src} as transitions in the replay buffer for training π is a surprisingly effective strategy when implemented with a couple of key design decisions we will introduce as we now describe the overall procedure.

Our method (summarized in Algorithm 7) is formulated as follows: We start by constructing a dataset $\mathcal{D}_{\text{src}}$ by rolling out the source policy π_{src} in $\mathcal{M}$. We then train our policy π_θ using off-policy RL with data sampled from both $\mathcal{D}_{\text{src}}$ and the new policy's replay buffer $\mathcal{D}$. We sample at a constant ratio ϕ from $\mathcal{D}_{\text{src}}$ (and $(1-\phi)$ from $\mathcal{D}$) to remain robust to the size of $\mathcal{D}_{\text{src}}$. We find through extensive experiments (please refer to the supplementary material) that this strategy and setting $\phi = 0.5$ strikes a good balance between

reliability and performance. Finally, the last component we find to be important (again, please refer to the supplementary material for ablations) is using a high *update-to-data ratio*, the ratio between policy updates (Algorithm 7, line 14) and data collection (line 10). This was also observed by Vecerík et al. [223], who used off-policy learning to incorporate human demonstrations to overcome sparse reward problems. They introduced L_2 regularization on the policy weights to accommodate this, whereas we find that the stochastic regularization used in [174] sufficient. Intuitively, taking more gradient updates may be important for effectively processing and incorporating the source policy data.

7.5 Learning Agile Locomotion Tasks with TWiRL

TWiRL is a robot learning system for acquiring complex locomotion skills by leveraging experience from related tasks or environments. In this section, we introduce concrete applications and how our framework can be applied to solve them.

7.5.1 Setup Details

We use the A1 robot from Unitree as our platform and build our simulation using PyBullet [73]. We define the state $\mathbf{s}_t$ to be a three-step history of the following features: root orientation, joint angles, base displacement, and previous actions. Our actions $\mathbf{a}_t$ are the target joint angles for the 12 actuated degrees-of-freedom. We do not adopt any additional engineered architectures for actions [30, 32, 47, 166] due to the generality of the agile motions that we aim to learn. The sensing-actuation control loop runs at a frequency of 20Hz.

The off-policy RL algorithm we use in our approach (outlined in Subsection 7.4.2) is a state-of-the-art off-policy actor-critic algorithm DroQ [174], a variant of SAC [72] that incorporates dropout regularization and layer normalization [176]. We use the open-sourced JAX [178] implementation from Smith et al. [38]. We report the average return and standard deviation across 3 runs unless otherwise stated.

7.5.2 Applications

7.5.2.1 Reward curriculum

While jumping over obstacles can drastically improve the mobility of a quadrupedal robot in cluttered environments, it is a particularly difficult skill for robots to master [255, 269]. For a successful jump, the robot must not only (i) reach the required speed to generate sufficient momentum to launch itself, but also (ii) dynamically adjust its step size depending on how soon it needs to jump. Then, while landing, the robot must learn to effectively (iii) leverage its high-bandwidth actuators to properly stabilize itself against

high-impulse contact forces. In contrast, the controllers most commonly studied for legged robots are quasi-static, with the robot being grounded by at least one foot at all times.

We design our jumping task so that the robot must contend with these three challenges by requiring the robot to jump consecutively over 20cm wide, 1cm tall 'hurdles' that are spaced randomly between 1.6 and 2.6 meters apart. As such, the robot should run up to each hurdle to jump over it, then land while continuing to run in order to execute the next jump at the right time. We define the reward function for this task with minimal shaping:

$$r(\mathbf{s}, \mathbf{a})^{\text{jump}} = r^{\text{rv}}(\mathbf{s}, \mathbf{a}) + r^{\text{o}}(\mathbf{s}, \mathbf{a}) + r^{\text{jv}}(\mathbf{s}, \mathbf{a}) + r^{\text{p}}(\mathbf{s}, \mathbf{a}), \tag{7.1}$$

where root velocity r^{rv} and progress r^{p} rewards encourage the robot to move forward, r^{o} penalizes undesirable orientations, and r^{jv} penalizes joint velocities. Please refer to the supplementary material for exact definitions. We terminate the episode if the robot collides with the hurdle. In order to learn the task, in addition to the features listed in Subsection 7.5.1, the observation space also contains a displacement vector between the robot's root position and the center of the hurdle, projected onto the ground plane.

As we see in Figure 7.3, training directly with the task reward in this case has two failure modes. First, we observe reward exploitation [87–89], where the robot steps over the hurdles unnaturally (please see the supplementary material to see videos of the behavior). In other runs, the robot greedily tried to move forward without hitting the hurdle by somersaulting (which was ultimately unsuccessful) and could not escape this local minimum. So, while this task is not well-suited to learning from scratch with RL, can this problem be solved by incorporating data from a policy that we *can* easily acquire? Prior work has come up with methods for imitating reference motions, enabling agents to learn naturalistic behaviors with shaped reward functions [57, 261–265]. Since these policies follow a fixed pattern (as prescribed by the reference motion), it is not robust to the variations required by the task we define of jumping over randomly placed hurdles. Nonetheless, we expect them to be useful to learn this more general task. As TWiRL is agnostic to the specific method used to produce the source policy, for our experiments, we choose to use the system proposed by Peng et al. [57] to learn a policy that imitates a leaping motion taken from the dataset provided by Zhang et al. [273]. Here, we collect 50k samples by rolling out this policy (trained in $\mathcal{M}_{\text{src}}$) in $\mathcal{M}_{\text{target}}$ to construct $\mathcal{D}_{\text{src}}$. Notably, *we do not initialize the task policy with the source policy is because of the mismatch in observations*—the motion imitation policy takes the fixed referenced trajectory as part of the input, while the task policy includes the distance to the next hurdle in observation for adapting to random hurdle locations. In Section 7.6, we demonstrate that the jumping task can be solved perfectly by incorporating this data via TWiRL.

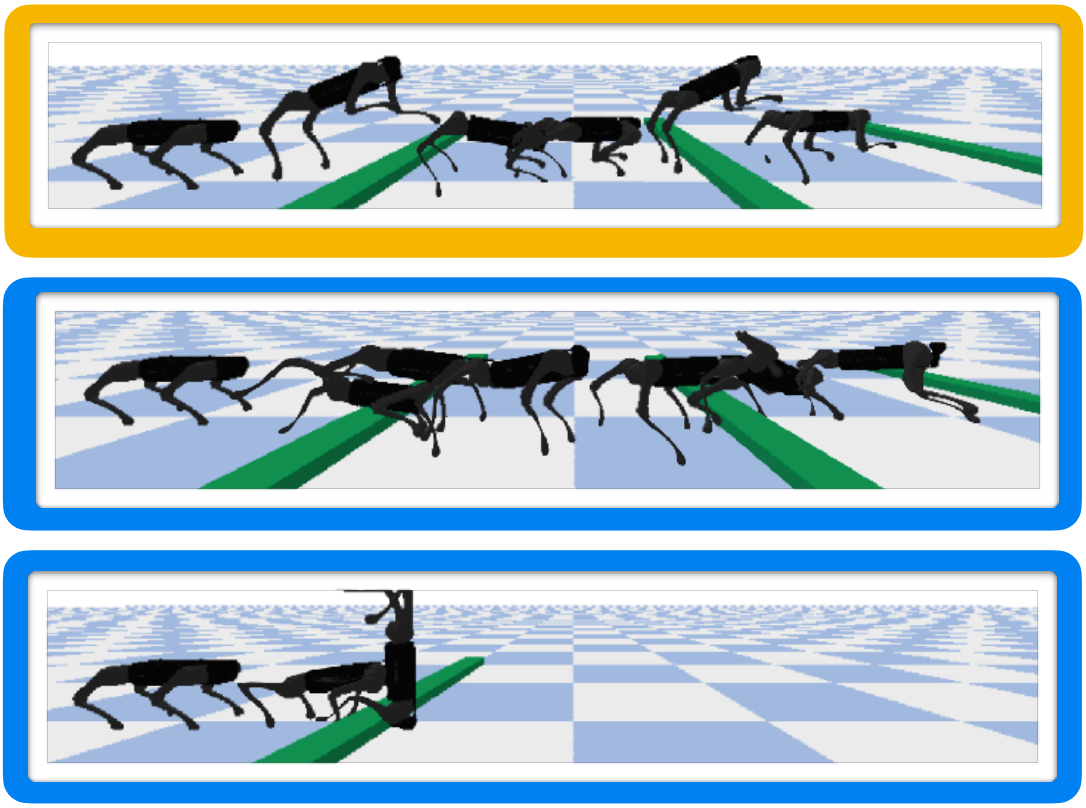


Figure 7.3: **(Comparing jumping skills learned with and without prior data)** Examples of our policy (outlined in yellow), which incorporates both online training and data from a motion imitation policy, compared to two policies (outlined in blue) trained from scratch with the same reward function. While naïvely optimizing for the task either exploits the simulator to learn an unnatural motion (middle) or fails completely (bottom), the policy trained by incorporating prior data exhibits a graceful jump.

7.5.2.2 Task curriculum

Bipedal walking requires less space and enables walking through narrow spaces. It can also accomplish tasks that are difficult for quadrupedal robots, such as climbing stairs and doing manipulation work with the front legs. However, the advantages also come with extra difficulties. One big challenge is balancing—the robot is inherently unstable on its hind legs, and maintaining balance while walking requires continuous adjustments to the body's center of mass, which is achieved through coordinated and precise control of the motors. Note that, unlike dedicated bipedal robots, the A1 has small rounded feet and a body shape that makes balancing on the hind legs more difficult.

In our experiments, we define a goal-conditioned navigation task in which the robot must acquire the agility to get up on its hind legs and walk to a desired location while

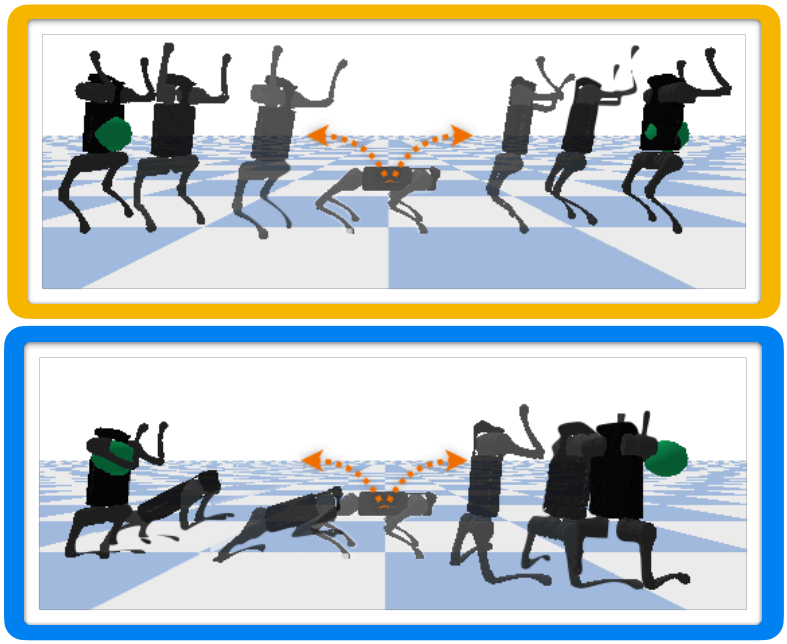


Figure 7.4: **(Comparing bipedal skills learned with and without prior data)** Examples of our policy (outlined in yellow), trained with data from a robot that can already stand on its hind legs, compared to a baseline policy (blue) trained from scratch. Without this added bias, the baseline policy learns to scoot toward the goal on its knees. Our policy gracefully kicks up to standing and navigates to the goal on 2 legs.

maintaining balance. We randomly sample these goals along a circle with a random radius, so as to train the robot to walk to a wide variety of goal locations. We again define a reward function with minimal shaping:

$$r(\mathbf{s}, \mathbf{a}) = r^{\mathrm{s}}(\mathbf{s}, \mathbf{a}) \cdot (1 + r^{\mathrm{f}}(\mathbf{s}, \mathbf{a}) + r^{\mathrm{d}}(\mathbf{s}, \mathbf{a})). \tag{7.2}$$

The 'stand' reward r_t^{s} encourages the robot's forward vector to be perpendicular to the ground plane and to maximize the heights of its front feet. The 'facing' reward r_t^{f} encourages the robot's belly to be pointed toward the goal, and the distance reward r_t^{d} gives the robot a constant bonus if its distance to the goal decreased during that step. For full details, please refer to the supplementary materials. The reward function encourages the robot to move towards the goal while maintaining the bipedal standing pose, as r_t^{s} acts as a gating function. For the task, the policy also receives the displacement vector, projected onto the ground plane, between itself and the goal.

As we show in Figure 7.4, training directly with the task reward again does not lead to successful learning. However, if we have a policy that has first learned just to stand, we can leverage experience from this policy with our method to make the bipedal navigation

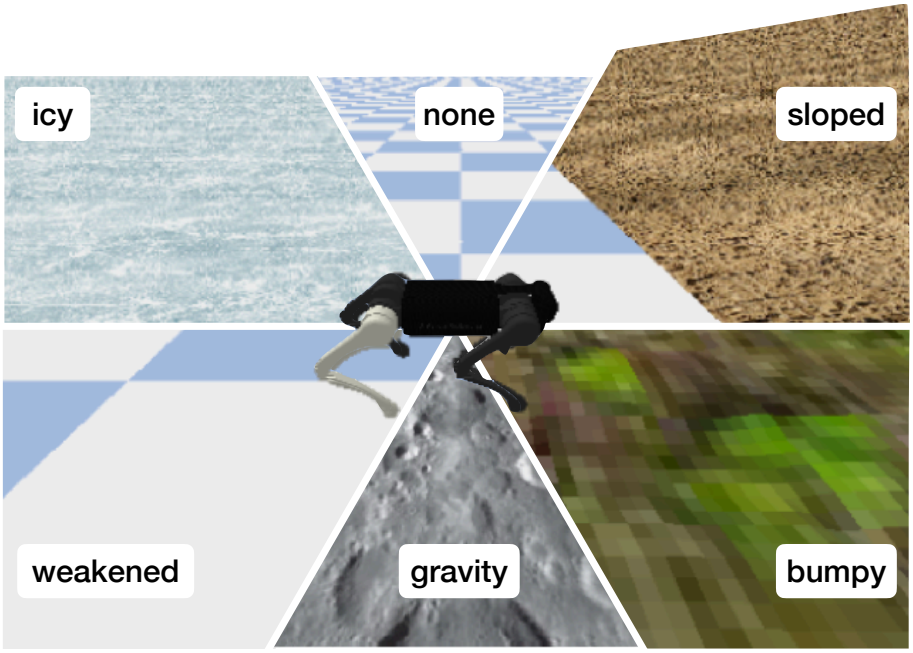


Figure 7.5: **(Simulated fine-tuning environments)** Illustrating the diverse environments to which we adapt the source policy (trained in the environment labeled 'none' to indicate no modification). In clockwise order: the default, non-randomized environment; a sloped terrain; bumpy terrain; a low-gravity environment; a stochastic environment simulating motor weakening; a simulated ice rink.

task practical to learn: We trained our source policy π_{src} with $r(\mathbf{s}, \mathbf{a}) = r^{\text{s}}(\mathbf{s}, \mathbf{a})$ just to get it to learn to stand on its hind legs. Then, after the robot has mastered getting up, it can move on to learning how to walk to the goal. Similarly to the jumping task, we collect 50k samples to populate $\mathcal{D}_{\text{src}}$ by rolling out this policy in $\mathcal{M}_{\text{target}}$.

7.5.2.3 Adapting skills to different environments

As the real world is complex and unpredictable, robots deployed in the wild will inevitably encounter situations for which they are not yet prepared and need to adapt their motor skills accordingly. A benefit of our framework which is illustrated in the two previous use cases is that it should allow us to learn skills with general, non-environment-specific reward functions, making them amenable to adapt to other environments. In our last example, we test our ability to adapt these skills to different environments using TWiRL. To do so, we set up a set of "sim-to-sim" transfer experiments, where we pre-train π_{src} in the same simulated environment and then transfer the policy in a variety of *other* simulated environments whose dynamics p differ from the pre-training setting. Specifically, we transfer under the conditions depicted in Figure 7.5 (for exact details please see the supplementary material). This setting is distinct from the other two in that it is possible

in this case to *also* transfer the policy network since only the dynamics are changing. Therefore, in these experiments, we will additionally study whether transferring the policy is helpful.

7.6 Results

We now present our experimental analysis of TWiRL. As a baseline, we evaluate directly applying RL to solve the tasks introduced in Section 7.5 from scratch. We then demonstrate how these tasks can be addressed more effectively by casting them as transfer problems for which we can readily acquire source policies—we show that incorporating off-policy data can (i) overcome exploration challenges, and (ii) bias the learning algorithm toward acquiring locomotion skills that are agile, robust, and safe for execution on physical hardware. Our experiments aim to answer the following concrete questions:

1. Can current RL methods handle the challenges of learning agile locomotion skills?
2. Can incorporating data from other policies using TWiRL enable the robot to learn these challenging tasks?
3. Can TWiRL policies be deployed in the real world?
4. Can we apply TWiRL to adapt to different dynamics?

To answer (1), we first try to train a policy from scratch on the jumping and bipedal locomotion tasks we defined in Subsection 7.5.2 (using the same underlying policy optimizer as TWiRL). We found that we were unable to train a policy to solve either task in this way (see the yellow curves in Figure 7.6). For the jumping task (the left plot), TWiRL (blue) consistently achieves $\mathbf{2}\times$ the return achieved by training for the task from scratch, which in addition to its poor average has *very* high variance across seeds. We found that the policies either could not discover the behavior necessary to jump over the hurdles, or learned to hobble over the hurdles but in a very unnatural manner (both behaviors pictured in Figure 7.3, blue). As shown in the second plot of Figure 7.6, learning the bipedal navigation task from scratch consistently converges to a local optimum. The robot learns to 'sit' while facing the goal (see Figure 7.8b), as this allows the robot to collect reward for its root orientation without having to accomplish the much harder task of getting up and walking. These experiments confirm that it is indeed unlikely for a robot trained from scratch to perform a complex task to acquire desirable behaviors.

To answer (2), we test whether using TWiRL to incorporate data from policies trained with different objectives effectively overcomes the challenges that prevent successful learning from scratch. As discussed in Section 7.5, for jumping, we use a policy trained to

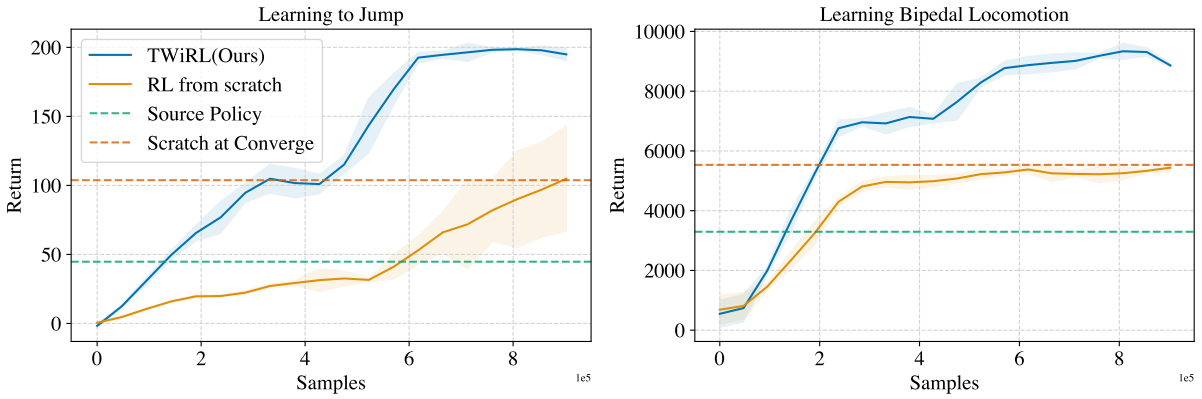


Figure 7.6: Learning curves for the jumping (left) and bipedal navigation (right) tasks comparing learning from scratch (solid yellow curve) to TWiRL (solid blue curve) using data from a source policy (dotted green line), shown to 1M steps. Since the policies from scratch had not yet converged for the jumping task, we gave an additional 2 million steps and visualized the average at convergence as well (yellow dotted line); we omit its variance for clarity. We see that in both cases TWiRL is able to far surpass the policies learned from scratch and the source policies.

imitate an animated leaping motion, and for bipedal navigation, a policy trained solely to stand up on its hind legs. For both tasks, the corresponding source policy is suboptimal when evaluated with respect to the target task—in the jumping task we consider variable spacing between hurdles, so following a prescribed trajectory will fail; in the bipedal navigation task, we consider a goal-conditioned task, so the one trained without a notion of this goal will not work. In fact, we quantify the quality of source policies with the dashed green lines in Figure 7.6 (averaged over 100 trials). However, we expect these behaviors to nonetheless provide useful information for learning the target task. We see that by incorporating this suboptimal data, TWiRL (solid blue curve) converges to an effective policy for both tasks in fewer than 1 million samples. Interestingly, we find that the resulting policy empirically also retains the naturalistic style exhibited by the source policy (see Figure 7.3 and Figure 7.4). This is not guaranteed since, unlike other works [90, 219, 221, 271], our objective does not specify any explicit requirement for the policy to resemble the source policy data. Instead, we simply provide examples and allow the value-based RL algorithm to discover the strategies that are useful within them. The advantage of this approach is that we give the robot as much freedom as possible to optimize for the desired task.

For (3), we evaluate our trained policies on a real A1 quadrupedal robot and find that they perform extremely well—we encourage the reader to see `https://sites.google.com/berkeley.edu/twirl`for all videos. The jumping policy tends to amble forward, very stably and almost nonchalantly, before *launching* itself over a hurdle. All four feet are

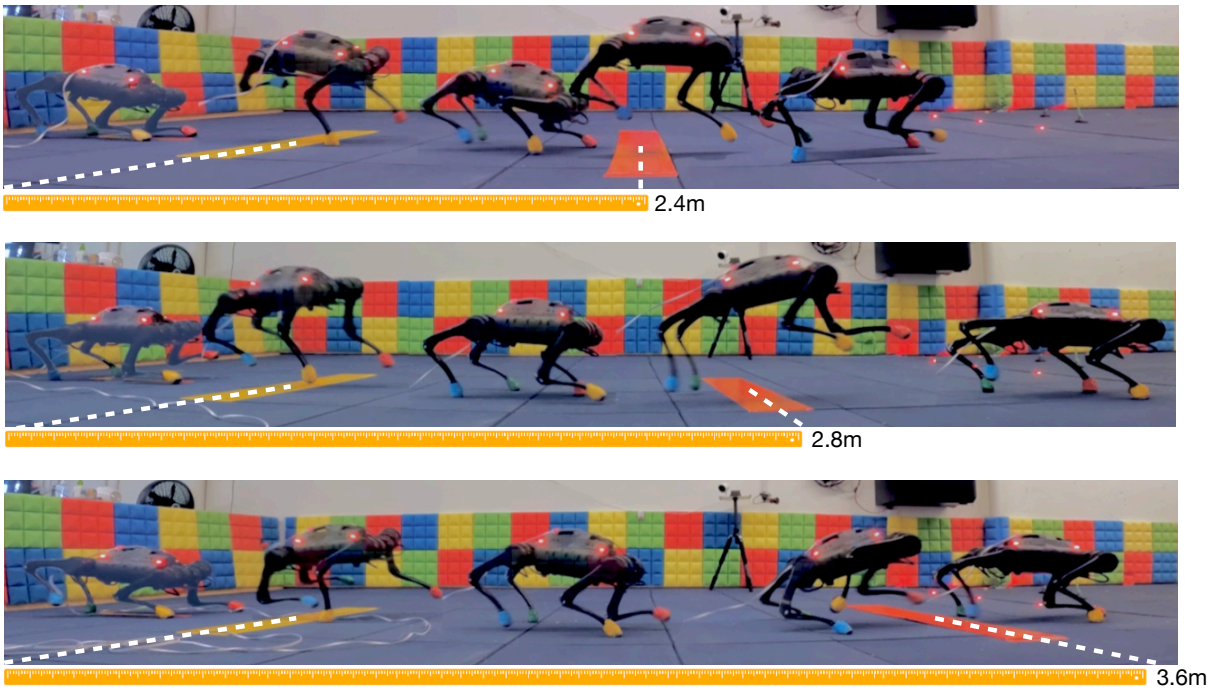
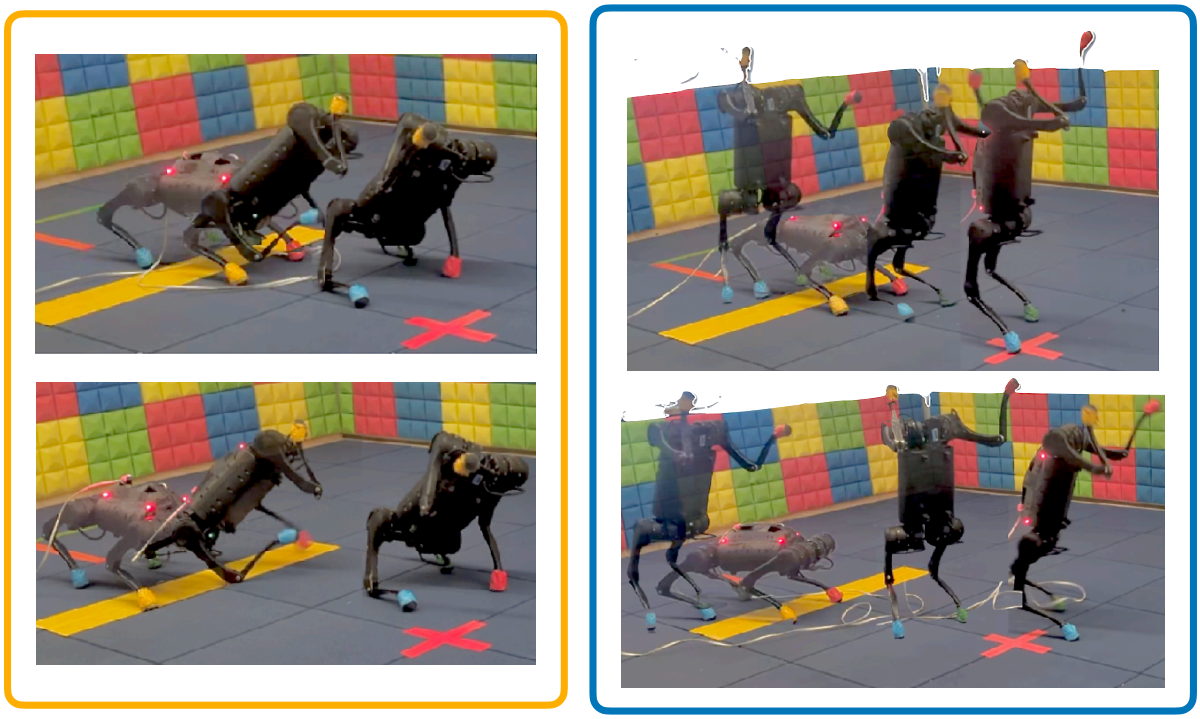


Figure 7.7: **(Real-world jumping rollouts)** Example successful rollouts from evaluating our jumping policy in the real world with different spacings of hurdles. From top to bottom the spacing is at 2.4m, 2.8m, and 3.6m, with success rates over 8 trials of **75%, 100%, and 75%**, respectively. We see that our policy exhibits the desired behavior (derived from the motion imitation policy) while being robust to task variation, but furthermore, it is able to be deployed on hardware. Videos of all evaluation trials can be found at `https://sites.google.com/berkeley.edu/twirl`.

simultaneously airborne before it reaches forward with its front legs to touch down, then steadies itself with its hind legs and continues forward. For jumping, we test 3 variations (corresponding to different hurdle placements) of the task, and we visualize the lengths and distances of these hurdles with colored tape on the ground (see Figure 7.7). Our jumping policy is able to consistently jump over both hurdles with variable spacing, land on its feet, and keep running. It does so whilst *carrying its entire body over the full length of the hurdle during one flight phase*. In contrast, the source policy is only able to jump up to one time before face-planting on the real robot, so we relegate these results to the supplementary material. For the bipedal navigation task, we place a goal (indicated by the red 'X' on the ground) about a meter from the robot's starting location (Figure 7.8). Our bipedal policy shoves itself onto its hind legs, walks forward, and approaches its target location, spending on average over 12 seconds upright. The A1 has small rubber spheres for feet, making this feat akin to balancing on tiptoe. The comparison 'bipedal' policy trained from scratch never attempts to stand up, only scoots toward the target on



(a) Examples a policy learned from scratch (left, yellow) and learned by TWiRL (right, blue) rolled out in the real world. Videos of all evaluation trials can be found at https://sites.google.com/berkeley.edu/twirl.

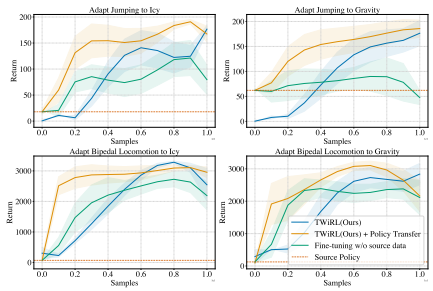
Metric	RL from Scratch	TWiRL
Standing Time	0.00 s	12.12 s
Min. Distance	0.49 m	0.14 m
Return	589	3127

(b) We report the time the robot remained standing, the minimum distance the robot got to the goal, and the return as defined by our task.

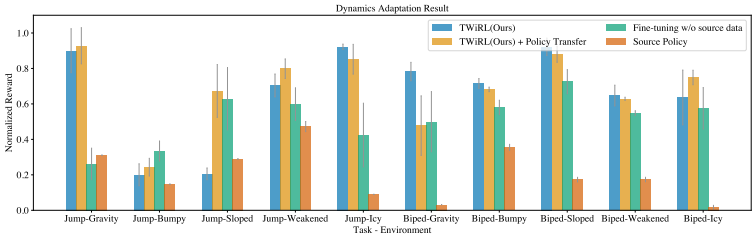
Figure 7.8: **(Real-world bipedal rollouts)** Comparison of our bipedal navigation policy to the baseline trained from scratch evaluated in the real world. Over 5 trials, our policy spent an average of 12 seconds upright and successfully approached the goal, whereas the baseline consistently scooted on its knees and never stood upright.

knees and one forepaw. Of course, though, the sim-to-real transfer is far from perfect, and we see this as an exciting opportunity for future work.

Lastly, to answer (4), we conduct experiments fine-tuning policies to 5 different environments listed in Section 7.5. Since the only shift in this setting is in the transition function (that is, the state space and task remain from the source policy) we repurpose



(a) Learning curves for adapting the jumping (top) and bipedal navigation (bottom) tasks to the low friction (left) and low gravity (right) environments.



(b) We report the performance for each task and environment combination after 100k samples' worth of fine-tuning with variants of TWiRL (meaning transferring the source policy data, blue and yellow) compared to discarding the data (green). For reference, we also report the zero-shot performance of the source policy in the target environment (orange).

Figure 7.9: **(Results adapting to different dynamics)** Evaluating TWiRL in adapting the jumping and bipedal navigation skills to 5 different environments (see Section 7.5) given a budget of 100k samples—corresponding to roughly 1.5 hours real-world time without any overhead. On the left, we show the learning curves for two environments in order to visualize the learning speed/dynamics and on the right, we summarize results for all environments. We see that in all cases, a variant of TWiRL is able to successfully allow the policies to adapt to new environments.

the replay buffer used to train the source policy to comprise $\mathcal{D}_{\text{src}}$ rather than having to collect additional data. For the same reason, we are able to test transferring the policy weights here as well. We show the learning curves for both tasks transferred to the simulated ice rink and modified gravity environments in Figure 7.9a. The most general version of our method wherein we do not assume we can transfer policy weights (blue) consistently enables the robot to re-learn the skill in a way that is tailored to the new environment in just 100k samples. Now, as expected, *additionally* transferring the policy weights (yellow) starts off in the new environment with more competency (than with vanilla TWiRL). We observe that both variants of TWiRL generally converge to similar performance, but perhaps surprisingly, discarding the policy weights actually sometimes surpasses the policy that was initialized with π_{src} after 100k samples. More importantly, though, we see that by comparing to a baseline method of transferring the policy and not the experience (green), incorporating the data from π_{src} makes an essential contribution to good performance.

7.7 Discussion and Future Work

We presented a framework for enabling quadrupedal robots to learn agile locomotion skills, including jumping and walking on the hind legs, by leveraging a transfer learning framework based on incorporating data from prior policies into training. We describe how the same basic transfer learning framework can make it possible to bootstrap more complex agile locomotion skills with simpler or more constrained ones, as well as enable

transfer of skills between environments. Our experimental evaluation shows that employing a curriculum with transfer learning can make it more practical to acquire effective policies for jumping and walking on the hind legs to a goal location than training from scratch, and that our proposed approach to transfer learning outperforms more naïve alternatives. Finally, we show that the agile skills that our method can learn can be used to control a real-world A1 quadrupedal robot, displaying a high level of agility.

While our work shows various ways to use transfer learning in service to learning more complex skills, it does not prescribe a single recipe that works in all cases—rather, we aim to illustrate a variety of ways in which the same transfer learning framework can be leveraged. This is also a limitation: the two behaviors we show (jumping and hind leg walking) use different curricula, with jumping employing motion imitation to bootstrap hurdle jumping and hind leg walking employing a hand-designed standing reward for pretraining. A more automated framework for curriculum learning that is based off of our approach could be an exciting direction to explore in future work. This could also enable a more diverse range of behaviors to be specified more easily, which could be particularly exciting as it could enable a broad range of agile behaviors for quadrupedal robots.

Chapter 8

Flexible Robotic Manipulation via Dense Language Grounding

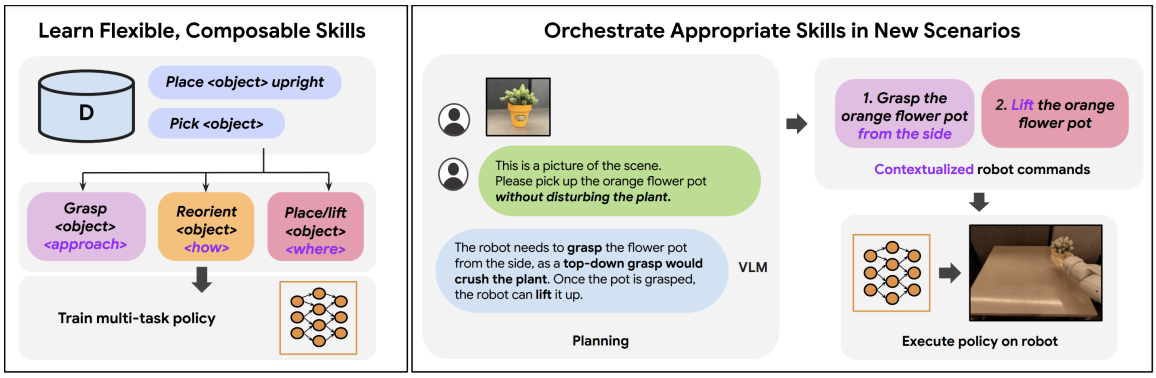


Figure 8.1: **(System diagram for STEER)** At training time, we re-annotate an offline dataset of diverse robot behaviors at training time, focusing on describing the primitive skills used to manipulate objects and, specifically, on annotating *how* the robot performed each skill. We then use this re-annotated dataset to train a language-conditioned low-level policy (RT-1 in our case). At inference time, when given a complex instruction like "pick up the flower pot *without disturbing the plant*", a high-level system (VLM or human) identifies the appropriate low-level skills and determines how to perform them. This emphasis on the "how" enables more contextual behavior.

8.1 Motivation

Consider the breadth of situations a human encounters on a daily basis, from pouring a cup of coffee in their kitchen to grabbing objects from a cluttered supply closet. Designing robot systems that can navigate these varied, nuanced scenarios is a major challenge, requiring systems that can adapt to complex and dynamic situations. This has led many

roboticists to explore learning-based solutions that may generalize better than hand-engineered ones. Imitation learning (IL) is a widely-used, data-driven approach that distills expert demonstrations into learned policies, enabling fine-grained manipulation of high-dimensional robot systems in the real world [9, 10], and has been shown to scale with more data in language-conditioned, multi-task [6–8, 274] and even multi-robot [11, 275, 276] regimes. While these works have shown remarkable promise, the resulting robot systems remain fairly limited to situations seen during training. And the span of these training scenarios is significantly narrower compared to those of other domains, such as vision and language, where large-scale supervised learning has excelled as real-world embodied data collection is significantly more expensive and bottlenecked by physical constraints.

Humans, on the other hand, can adapt to very complex situations without any previous first-hand experience. We exhibit 'common sense' generalization—using our inherent understanding of complex, high-level concepts like object affordances, intuitive physics, and compositionality to adapt our past experiences intelligently as new situations arise. This deliberate, analytical thinking has been termed 'System 2' processing, in contrast to reactive 'System 1', reflexive low-level behaviors that are less cognitively demanding but equally essential for our behavior [277]. Emulating this blend of reasoning and reflex in robotic systems is challenging, and various approaches have been developed to bridge the gap. One notable example is SayCan [278], which leverages a large language model (LLM) to plan over and sequence learned policies to perform long-horizon tasks. SayCan compensates for the LLM's lack of direct physical grounding by using the policies' value functions to assess feasibility. Moreover, this approach is limited to sequencing the original tasks demonstrated, while many realistic tasks require finer, more nuanced control of low-level policies (as illustrated in Figure 8.1, right side). Subsequent works have focused on enabling LLMs or VLMs to interface at a finer granularity with pre-programmed System 1 behaviors through representations like code [279] or semantic keypoints [280]. Another strategy that has emerged is to fine-tune VLMs on embodied data [8, 274], drawing on web-scale pre-trained representations for robot control. Notably, these approaches focus on optimizing the high-level module to make the best use of a fixed set of skills, which still constrains their adaptability in unstructured, novel scenarios. Rather than modifying the System 2 reasoning layer, we pursue an orthogonal direction by focusing on making System 1 policies more flexible and responsive to high-level guidance. By designing adaptable System 1 policies, we enable seamless interaction with a fixed System 2 module—such as human or VLM-based reasoning. This adaptability allows System 1 to be dynamically adjusted in response to high-level instructions, ultimately broadening the range of tasks the system can perform and its capacity for robust, generalizable behavior.

We present STEER: Structured Training for EmbodiEd Reasoning, an approach for training low-level reactive policies that can be flexibly steered or directed by a higher-level

reasoner, such as a human or VLM. Our key insight for enabling this is producing *dense language annotations* of the collected robot data, and training a conditional policy on granular language instructions. This policy can then be conditioned on each part of a plan generated from a high-level model (VLM/LLM), giving a combination of the respective strengths of System 1 and System 2 processes. Furthermore, this can enable adapting to new situations that require synthesizing behaviors that are not explicitly demonstrated during training. We instantiate this system using existing real-world datasets, proposing a simple automated labeling pipeline based on proprioceptive observations to extract basic object-centric manipulation skills and distill them into a low-level policy. We then propose a strategy for using a VLM to produce language directions for the low-level policy. Importantly, we show that this enables us to repurpose skills in the robot dataset in a meaningful manner at test time to handle a new situation autonomously. In summary, the contributions presented in this chapter are as follows:

- We introduce STEER, a method that augments robot demonstration datasets with descriptive functional language annotations comprising of grasp-centric and rotation-based primitive components.
- We show that STEER enables training low-level robot policies which are significantly more flexible and steerable than prior imitation learning methods, enabling humans or pre-trained VLMs to direct low-level policies for generalizing to novel everyday manipulation tasks.

8.2 Related Work

Imitation Learning. Imitation learning (IL) has emerged as the most popular paradigm for training real-world manipulation policies [6, 10, 281]; however, deploying these models in unstructured scenarios remains a challenge. Robot policies trained on human-collected demonstration data can have trouble adapting to "out-of-distribution" scenarios where demonstrations are sparse [282]. This is fundamentally due to the expensive-to-collect small scale robot data, in comparison to the web-scale text and image datasets for training today's foundation models [283–285]. Due to practical constraints on obtaining more robot data, a large body of work has explored using text and vision foundation models to improve generalization in robotics from existing datasets. This includes expanding IL policies to open-world object grasping by using open-vocabulary object detection [286] and relabeling episode-level instructions via CLIP [287] or other foundation models [288, 289]. Our work is closely related to aforementioned works in dataset relabeling [287–289]: our framework expands robot capabilities by relabeling different behavior modes in *existing* heterogeneous robot demo datasets to train more effective policies.

Policy Conditioning for Generalization. Prior works have sought to enable test-time generalization by exploring expressive modalities for policy conditioning, such as goal target poses [290], goal images [291, 292], trajectories [293, 294], code [279], or combinations of vision and language [295, 296]. However, while these modalities all show promise in action generalization, it is challenging for off-the-shelf high-level reasoning systems like VLMs or humans to plan over these creative modalities; natural language remains the main modality utilized in complex planning by state-of-the-art LLMs and VLMs. Thus, STEER focuses on improving language-conditioned action prediction, by studying *granular instructions* such as "grasp from the side," or "lift up," that can be easily composed at test-time.

Skill Learning. There is extensive research in utilizing learned 'skills' to accelerate learning new tasks by exploring with temporally-extended, semantically meaningful action sequences [290, 297–306]. These works often use a hierarchical approach, where a high-level policy is learned through interaction with reinforcement learning through environment interaction to compose the learned skills [301, 304–309]. EXTRACT [306] in particular uses VLMs to label skills from offline data to enable learning new tasks. Our work also exploits the intuition about skills being transferable to synthesize new behaviors. However, we use common-sense reasoning in off-the-shelf VLMs to choose appropriate skills for the situation without training a separate policy.

Affordances. Our work leverages VLMs' common-sense reasoning capabilities to plan for longer-horizon tasks and provide strategies for the robot to approach novel configurations of objects. The model does this by reasoning about how humans would approach tasks from visual image input. Prior work investigates how to represent human priors for how to act in scenes using affordances [310, 311] in image space, and even shown how to deploy these on real robot systems for guiding exploration [312, 313]. While effective, these affordances are often represented in the form of keypoint coordinates in pixel space, and make particular assumptions on the kinds of tasks to be performed. Our approach can be thought of reasoning about such affordances in language, which makes it amenable for off-the-shelf VLMs or humans to naturally interact with and opens the door to express more sophisticated descriptions than in visual space.

8.3 System Design

We present STEER, a robot learning framework that aims to expand the capabilities of a robot trained on a set of expert demonstrations by extracting flexible skills from existing datasets, then relying on a module with strong reasoning capabilities to orchestrate the skills intelligently. Our system consists of two main components: low-level 'System 1' skill-training and high-level 'System 2' high-level reasoning. In this section, we describe design decisions for acquiring and then integrating them into a practical end-to-end system.

8.3.1 Learning Flexible, Composable Manipulation Skills

Our goal is to extract skills that can be easily reasoned about and composed by either humans of foundation models. Thus, we aim for skills that are language-indexed and object-centric, allowing a foundation model with knowledge about how the state of an object should evolve to accomplish a certain task to be able to steer despite not having direct motor control capabilities. To achieve this, we densely annotate existing datasets, then train a language-conditioned RT-1 [7] policy with these segmented and relabeled instructions.

Our key idea is to break down basic composable skills into *semantically identifiable* categories that can be associated with a language description. As intuition, many have observed that human-collected behavior data is challenging to learn from in part due to different data collectors having different 'styles' or strategies [314]. For example, human driving styles—aggressive, cautious, smooth, or jerky—are highly variable. Prior works have dealt with this heterogeneity by using latent variables to explain modes [315], new algorithms [316], or more expressive generative models [10, 317]. We expose these different styles as adjustable parameters, allowing robots to flexibly adapt their behavior. We focus on shared, object-relational skills such as grasping, lifting, placing, and rotating. These skills, originally demonstrated using templates like `pick <object>, move <object1> near <object2>, knock <object>, place <object> upright`, can be executed with varying strategies. We identify the following key factors:

Grasp Angle. Objects can be grasped in multiple stable positions, and the particular way indeed impacts the ability to perform downstream tasks. However, grasp positions are rarely prescribed (and therefore labeled), as they are often implicit. We hypothesize that controlling grasp angle can improve task composition and adaptability. We use a simple approach to label the grasp approach by manually labeling a relatively small set of 'anchor' grasp poses. We then label an arbitrary grasp with the label of its nearest neighbor 'anchor' pose as measured by cosine similarity.

We represent a grasp pose as a 3D unit vector by rotating $[0, 1, 0]$ by the wrist quaternion and we identify the time of a grasp where the gripper changed from fully open to fully closed. To define and label the anchor poses, we took 3D unit vectors that are linear combinations of the elementary 3D basis vectors (resulting in 27 directions). We then clustered 1000 grasps from our dataset and visualized the clusters in order to label them. This only requires visualizing and labeling roughly 20 clusters, but then can be used to automatically label the entire dataset of 70K demonstrations. In the grasp data, we identify three distinct modes via inspecting the grasp images: top-down grasps, side grasps, and diagonal grasps (visualized in Figure 8.2). The sub-trajectory starting from the beginning of the demonstration to the time of the grasp identified is relabeled to `grasp the <object> in a <grasp approach>`, where `<object>` is from the original instruction

and `<grasp approach>` is from the anchor's label.

Reorientation. Another mode of behavior identified in the dataset is of reorienting objects. In order to identify and label these reorientations, we first label the wrist orientation for every timestep where the gripper is fully closed. Then if the gripper orientation switches between two of the modes (as labeled in **Grasp Angle**), we label the sub-trajectory preceding it as `reorient the <object> <direction>`, where `<object>` again is from the original instruction and `<direction>` indicates whether the object is rotated from upright to horizontal or vice versa.

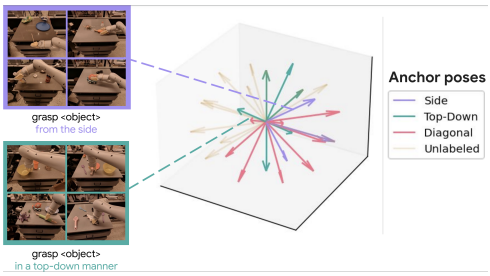


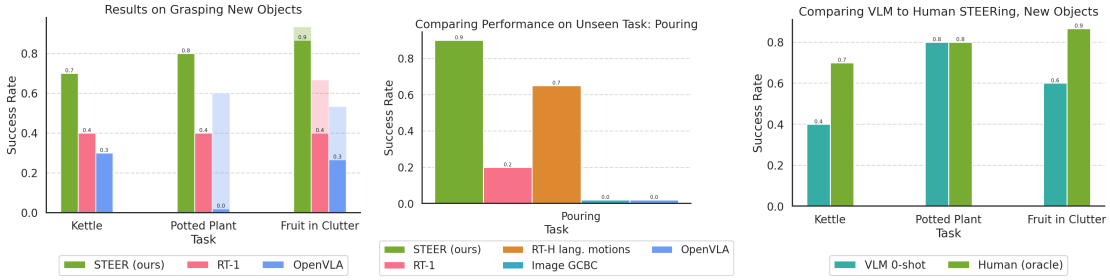
Figure 8.2: **(Visualization of grasp angle labels)** Anchor vectors and their semantic labels. Purple, green, and pink vectors represent side, top-down, and diagonal grasps.

Lifting/Placing. Complementing grasping, we label whether the object was lifted or placed at the end of completing the original task. Lifting allows the robot to continue in-hand manipulation without setting the object down or provides the ability to gain additional clearance. If the object is still held at the end of the episode and the gripper moves vertically upward (common for originally-labeled `pick <object>` episodes), we label the final sub-trajectory as `hold and lift the <object>`. If not, similar to identifying grasps, we identify the time of placing using the gripper state and label this sub-trajectory as `place the <object>`. As this is separate from reorientation, placing implies setting the object down while maintaining its orientation.

8.3.2 Orchestrating Learned Skills

A key capability of the System 2 component is its ability to reason about the visual observation of the scene, the task description, and the robot's low-level skills to effectively select and sequence appropriate actions. While a human can perform this reasoning, we also demonstrate how it can be automated using a VLM. Our automated System 2 component is implemented as a code-writing VLM agent, enabling it to autonomously execute verbalized plans without additional modules or a human in the loop. To facilitate this, we define an API for the action primitives accessible by the VLM to interface with the System 1, reactive low-level RT-1 policy skills as described in Subsection 8.3.1. The API is based on translating the language commands into a simple API that the VLM agent can access. This breakdown is based on *what* the robot should do and *how* to do it. Each primitive skill (i.e. grasping, rotating, lifting, placing) is represented by a function with a keyword argument modifying *how* that primitive is accomplished (i.e. `grasp(object, "top-down")`). Internally, the API translates this code into the corresponding natural language the RT-1 policy was trained on. Following Arenas et al. [318], we use a system

prompt to tell the VLM to control its physical embodiment through code, then provide the robot's visual observation of the scene and a description of the high-level task. One benefit of this design is its modularity and interpretability. The prompt can be modified to tailor behavior according to the specific situation the robot is in, the generated plans can be viewed before execution. The exact system prompt we use in all experiments and example outputs produced by the model can be found at https://lauramsmith.github.io/steer.



(a) Grasping results. Full successes are in solid colors. Partial successes (defined as grasping the object with undesired side effects like disturbing the plant or other objects) are in light colors. For each method, we do 20 trials for `Kettle`, 10 trials for `Potted Plant`, and 15 trials/configurations of `Fruit in Clutter`.

(b) New task results. We run each method 10 times, comparing the upper bound on performance of the low-level capabilities afforded by each model to perform the task. Therefore, we use closed-loop human guidance for language-instructed models and granular goal images for the image-conditioned policy.

(c) We report the success rate of a VLM controlling the robot compared to the human operator and find that the VLM produces reasonable plans but there is low-level policy failure due to sensitivity of the low-level policy to language inputs.

Figure 8.3: **(Real-world results)** Results on grasping in unseen scenarios and performing a new task, with human or VLM guidance. We find that by having access to and being able to reason about extracted low-level strategies enables higher success in OOD scenarios than the baseline RT-1 model and a state-of-the-art VLA.

8.4 Experiments

To understand the efficacy afforded by multiple strategies extracted and learned from the offline data, we test whether our model enables more effective grasping in unseen scenarios. We test this by manipulating objects that do not appear in the offline dataset *and* require specific grasp strategies to succeed. We then test STEER's ability to perform new behaviors—requiring complex reasoning *and* reliable motor control. This is first demonstrated by having a human expert create a plan for STEER, to show our design decisions lead to composable primitives. We then leverage a VLM to automate the planning. We aim to answer the following concrete questions through our experiments:

1. Does learning multiple modes of behaviors used to solve a task improve the adaptability of a robotic manipulation system to novel situations?
2. Can combining extracted skills from heterogeneous human demos enable new tasks?
3. To what degree can a state-of-the-art VLM plan orchestrate these skills autonomously?

8.4.1 Experimental Setup

We use a mobile manipulator with a 7 DoF arm, a two-fingered gripper, and a mobile base, as used in RT-1 [7]. We target a tabletop manipulation environment, where objects on a counter need to be moved or arranged according to natural language instruction. We use the multi-task (6 semantic categories) demonstration dataset used in RT-1 [7] (70K demonstrations) and the dataset of grasping-only data featured in MOO [286] (15K demonstrations). As we discuss in Section 8.3, the exact architectures of the low-level (System 1) and high-level (System 2) components can vary, as long as the high-level component can reason over skills expressed in language. We choose RT-1 [7] for our System 1 component and Gemini 1.5 Pro [319] in VLM experiments. Videos and setup details are available at `https://lauramsmith.github.io/steer`.

Figure 8.4: **(Sample initial conditions)** for the new object-grasping scenarios evaluated. (top left) a kettle with a handle extending above it, (top right) a potted plant, (bottom) 2/15 scenes for `Fruit in Clutter`. The kettle should be grasped over top. In order to avoid disturbing the plant, the flower pot should be grasped around its body. Lastly, the fruits should be grasped while avoiding knocking over the other objects in the scene.

8.4.2 Improving Test-Time Adaptability

Setup. One of our central hypotheses is that explicitly extracting, labeling, and training expressive primitives from heterogeneous demonstrations affords our system with improved robustness. When faced with unfamiliar situations at test time, we expect some skills to generalize better or be more suitable than others. To test this, we present the robot with three unseen object-grasping scenarios (sample initial conditions shown in Figure 8.4): a kettle with a handle extending above it, a potted plant, and grasping a fruit out of clutter. We choose unseen test objects specifically to simulate challenging real-world settings in which naïve grasping without a strategy is unlikely to succeed.

We first compare STEER against the baseline RT-1 [7] model, with the same architecture,

trained on the same demo data as STEER, but with the *original language instructions*. We condition STEER on the templated language it is trained on, with the grasp strategy chosen by a human in these experiments. We condition RT-1 on the templated language it is trained on, i.e. "pick ¡object¿". For STEER and RT-1, we train with a 50/50 split of the RT-1 and MOO datasets since MOO comprises diverse grasping-only data. We then compare to OpenVLA [11], a model that fine-tunes a VLM [283] on robot data from the Open X-Embodiment dataset [320]. The OXE dataset includes the same demo data that RT-1 and STEER were trained on, in addition to data from various other robots and tasks. This is meant to test whether fine-tuning a VLM on robot data is sufficient to elicit the desired reasoning and downstream execution capabilities for these tasks as opposed to explicit re-annotation and direction by a high-level module. We condition OpenVLA on the same language, i.e., "pick orange flower pot without disturbing the plant" or "pick apple while avoiding the other objects" that we also provide our VLM to sufficiently define the task in Subsection 8.4.4.

Results. We report the success rates in Figure 8.3a. RT-1 occasionally succeeds, but exhibits different strategies and we observe that failures are often caused by a sub-optimal approach. For example, when approaching the potted plant without a direct side grasp, the pot is prone to falling or grasping the plant leaves. OpenVLA performed similarly to RT-1, demonstrating that additional web data does not necessarily lead to sufficiently strong embodied reasoning about *how* to grasp in a new scenario where a particular approach is evidently necessary. For example, we find that OpenVLA often picks the potted plant up, but does not respect the language instruction of picking up the flower pot *without disturbing the plant* and grasps from above around the plant leaves. We further test whether it is *possible* with human guidance to steer RT-1 and OpenVLA to grasp differently through the same language prompting as STEER. As shown in Figure 8.5, STEER clearly changes its grasp strategy based on language prompting, whereas without re-annotation, neither OpenVLA nor RT-1 adjusts its behavior with more descriptive conditioning. This highlights that decomposing the grasp strategies and exploiting the most suitable one as we do in STEER is necessary in this case to coax a model to generalize correctly.

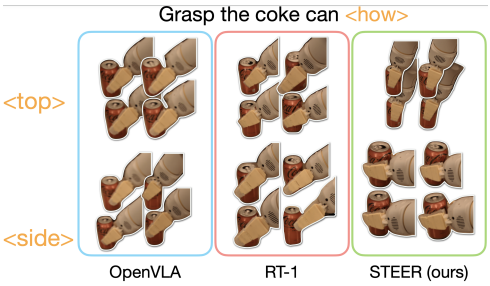


Figure 8.5: **(Grasp steerability)** of OpenVLA, RT-1, and STEER. We test the steerability to grasp an object in different ways that would be appropriate for different, unseen tasks, e.g., in order to pour out of the Coke can, the robot should grasp the can around its body. When prompted to "Grasp the Coke can" from the top (top row) versus the side (bottom row), models without dense annotation show no perceivable change, while our densely labeled model adjusts its behavior, enabling new downstream tasks.

8.4.3 Performing Novel Behaviors

Setup. Having demonstrated that STEER learns more flexible skill primitives, we study how well we can engineer behavior for a new everyday task without collecting new demonstrations or additional fine-tuning. To test this, we consider an everyday task, pouring, that is out of the distribution of demonstrated tasks but should be achievable with the motions that exist in the data. Pouring requires grasping the cup from the side, such that the robot can easily tilt the cup once lifted—avoiding singularities or spilling onto the robot—then setting the cup back down onto the table upright.

We perform an extensive evaluation on the pouring task, comparing against the **best-case** version of each baseline and comparison. Each method exposes a different control interface—for example, RT-1 is commanded with the natural language instructions it has been trained on, while a BC policy conditioned on goal images is commanded by giving images of the sub-goals required to reach the desired end state. For each method, if human assistance is used, the human coaxes the model to perform the task by providing what they deem to be the best command that method supports, in a closed-loop fashion. The human is not allowed to move the objects or robot on their own, and a trial is halted if the robot reaches an irrecoverable state (e.g. objects fall off the table). For this task, we train RT-1 and STEER with a mixing ratio proportional to the respective dataset sizes (roughly 85% RT-1 dataset 15% MOO dataset) as we are not testing generalization to new *objects*, rather maneuvering seen objects in new ways. We compare to the following 4 baselines and prior methods:

1. RT-1 [7], which acts as our baseline set of action primitives given by the original tasks in the datasets.
2. Language motions from RT-H [321], defined by narrating end-effector movement to give language like `move arm left` and `rotate arm right`.
3. A goal-image conditioned variant of RT-1. This tests whether language is a better abstraction than goal images. For this comparison, we first perform a demonstration of the new task, then run a goal-image conditioned policy passing images from that demonstration as subgoals at the granularity of our extracted skills.
4. OpenVLA [11], to compare to a state-of-the-art model pre-trained on web data and fine-tuned on robot data.

Results. Human orchestration with a STEER policy achieves a 90% success rate on pouring as compared to 70% with a policy trained with language motions from RT-H (Figure 8.3b). In comparison, baseline RT-1 cannot complete the task because it is not trained to reorient objects. Instead, it is trained to knock over objects and place objects

upright. If the policy successfully places the cup back upright after knocking, though, we consider the trial a half-success since knocking the cup onto the table is not a desirable pouring motion. The goal image baseline, despite having demonstration subgoal images from the same starting positions, fails to perform the task successfully. We observe that the policy is brittle and appears to match the arm pose in the subgoals, rather than manipulate the object state as prescribed by the goal image. OpenVLA, despite having access to the same underlying robot demonstration data and VLM pre-training, does not generalize to the new motion by stitching together the appropriate motions. Instead, we observe that it often knocks over the cup. Despite the very dense language motion instructions proposed by RT-H achieving a high success rate, orchestration is significantly more cumbersome as it requires tight closed-loop guidance. This is evidenced by requiring on average 15 instructions of the type "move arm forward and left" done in-the-loop as opposed to 5 simple commands that can be executed open-loop by STEER (i.e., "grasp pink cup from the side", "lift pink cup", "reorient the pink cup to be horizontal", "reorient the pink cup to be vertical", "place the pink cup").

8.4.4 VLM Orchestration

Now, we test whether a *VLM* can effectively select or sequence appropriate skills afforded by STEER by reasoning about the context, in the visual observation and task description, as well as the skills exposed through the API *without any examples* (i.e. 0-shot). For these experiments, we compare to human orchestration of the same STEER policy as an upper bound on the performance. For each trial, we query the VLM with the initial scene, task description, and maintain the same system prompt.

Seen task, new scenarios. We see that the VLM successfully produces the same high-level plans as the human expert very reliably for the grasping tasks. However, as shown in Figure 8.3c we see that there is a degradation in end-to-end task performance compared to human orchestration when executing the code produced by the VLM, and we analyze these failures. For the kettle picking task, we note that the low-level policy appears to be sensitive to the specific naming of objects. That is, the VLM often produced code to grasp the 'black and white kettle' from the top instead of grasping the 'black and white object' from the top, and with further analysis find that this instruction has a noticeable degradation across all low-level language-conditioned policies. So, while the VLM reasonably commands the policy to grasp from above, the low-level policy is less reliable. We expect this to be improved with denser annotation or augmentation on the entity-level, whereas STEER is concerned with the motion-level. For the `Fruit in Clutter` grasping task, the VLM did not always command the appropriate action and we suspect that similar object naming references ('red apple' instead of 'apple') impact the low-level policy performance.

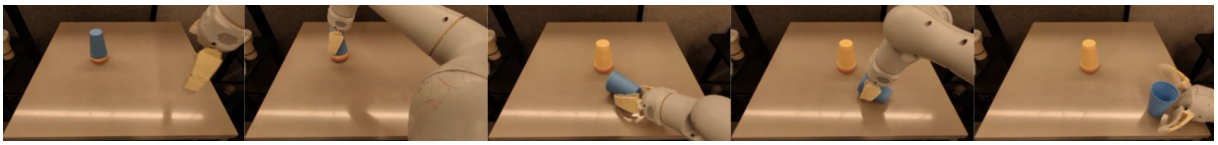


Figure 8.6: **(STEER solving a new complex manipulation task leveraging steerable low-level skills)**

Seen objects, new task. We observe that without any examples, the VLM correctly identifies the grasp location to pour from the cup,in a manner that a human would perform the task. It then recognizes that it must reorient it, then reorient it back in order to place it back upright on the table. A common failure mode is that the VLM misunderstands that 90 degree rotation does not invert the cup. However, this artifact luckily does not affect performance on *this* task. The VLM succeeded in 6/10 trials for zero-shot pouring.

8.4.5 Additional Experiments

Self-improvement with in-context learning. For the pouring task, the VLM orchestrated policy succeeded in 6/10 trials. We tested whether feeding the VLM-generated programs that resulted in on-robot success as examples in the prompt would lead to a higher success rate. Indeed, with the self-generated in-context examples, the newly generated programs succeeded in 8/10 trials for the same pouring task, improving upon the 6/10 in 0-shot. This only requires human labels for task success and no model fine-tuning.

Cup unstacking and flipping. We also test our ability to use STEER to perform a longer-horizon task of unstacking and reorienting a cup upright, so as to be able to dispense a drink. This task also requires grasping *with intention* for the future object reorientation steps. We demonstrate that we can guide STEER to perform this new task (see Figure 8.6).

8.5 Discussion

We presented STEER, a robotic system that can follow natural language instructions for manipulation. Rather than collecting data for new tasks, STEER uses a novel methodology for relabeling *existing* data with flexible and composable manipulation primitives. This relabeled data is used to train a small language-conditioned policy, which can be controlled using either a high-level VLM agent or a human. STEER can be *steered* to perform specific manipulation tasks, which in conjunction with the commonsense-reasoning afforded by a VLM agent can perform intelligent multi-step manipulation without ever

collecting new data. We report that our simple recipe can outperform large, end-to-end models like OpenVLA (despite using a 100× smaller model and data). Since the performance of STEER is driven by our re-annotation of behavior modes, this suggests existing robot datasets could also be re-annotated to produce more steerable action primitives, with little changes to the model architecture and training pipelines. In the future, it would be useful to scale up the discovery and labeling of dataset attributes, investigate automatic relabeling, and more directly optimize annotations to maximize the high-level agent's skill composability.

Chapter 9

Conclusion

In this thesis, we considered the problem of building robots that can autonomously navigate real-world complexity by enabling them to continuously learn and adapt as they encounter new, unseen situations. In Part I, we discussed practical challenges that preclude scalable real-world learning: being able to operate without human supervision or intervention, not relying on external sensors, and obtaining supervision for training. In Part II, we pushed this paradigm further by learning complex motor skills in minutes completely in the real world and provided perspectives on exploration designed *for* the real-world challenges of efficiency, safety, and non-stationarity. Lastly, in Part III, we looked towards more complex skills by incorporating priors: offline data or reasoning using foundation models. In this chapter, we will conclude by discussing takeaways from this body of work, remaining limitations, and avenues for future work.

First, we will discuss some reflections from the primary results in training locomotion skills in the real world. We began by demonstrating adaptation of motion imitation skills, trained in simulation, using about an hour or two of autonomously collected real-world data. The critical pieces introduced in this work were the abilities to autonomously detect and then recover from failures and to train successfully in completely unstructured, out-of-lab settings. These form the basic structure of a robot system that can almost fully autonomously improve during deployment. With these tools, we knew it was possible to have a legged robot learn high dimensional control from very noisy, real-world signals in outdoor settings.

By delving further into the question of learning efficiency, we found that one of the largest indicators of training speed is the exploration space. That is, having the robot find the optimal solution starting from a completely unconstrained action space is extremely expensive. Whereas, if the robot explores in the neighborhood of a good solution, learning can happen extremely quickly. This allowed us to eliminate the need for simulated pre-training to enable learning to walk completely from scratch in a few minutes, specifically

because inducing a 'good' exploration space for simple walking is straightforward (see Chapter 4 for more discussion on this topic). To lift the manually-defined constraints on the robot's exploration, we thus extended with an automated method in Chapter 5. This offers another perspective on the work in Chapter 6 as a data-driven way to shape exploration using offline data. This perspective is expanded upon in Chapter 7, i.e., training from scratch often completely fails but the inclusion of offline data very quickly shapes the exploration, and as a result, allows for extremely efficient learning. We will revisit this perspective on using prior data as a practical avenue to accelerate real-world RL next in the context of other robot applications.

While these demonstrations provide evidence for the efficacy of real-world RL for physical robot systems, they have been limited to learning locomotion skills. In what ways do these insights transfer to other domains, such as robotic manipulation or driving? One common thread is as mentioned above: that adapting to a real-world situation through practice can be made very efficient if the exploration is 'shaped', making the real-world training feasible. A possible response to this problem is to pre-train in simulation as in Chapter 2, but simulating training for general robotic tasks is not as straightforward as for locomotion skills. For example, generating training terrains—slopes, stairs, rugged terrain, gaps are fairly simple to programmatically generate, and it is thus scalable. Additionally, rewards are somewhat straightforward to define for locomotion tasks: imitating motion clips, which admit a natural dense reward function, or using a velocity-based reward with many pose or energy-based regularizers. In comparison, it is not nearly as straightforward to programmatically generate millions of meaningful assets to make training environments or define rewards for manipulation (see Chapter 3).

Therefore, another possible response that does not rely on simulation is in line with that of Chapter 6: including mixed-quality prior data in this off-policy RL framework, which empirically shapes early exploration and makes learning for the desired skill efficient enough for real-world constraints. In Chapter 7, we argued that for robotics applications it is often relatively easy to acquire some real-world data that is related to a desired task and provided concrete examples for highly agile locomotion. It is quite straightforward to extend this argument to other domains, and there are real-world results to demonstrate this framework with appropriate 'source controllers'. For example, Mendonca et al. [322] demonstrate efficient learning of mobile manipulation skills in the real world with RLPD by using simple scripted policies to collect initial exploratory data to bootstrap off of, as well as multi-task learning to facilitate autonomous training. Stachowicz et al. [323] demonstrate learning of fast driving policies via RLPD in various natural environments by bootstrapping off suboptimal human demonstrations. Luo et al. [324] offer a comprehensive software package that extends RLPD for training real-world, vision-based manipulation tasks, leveraging easy-to-provide human demonstrations. This work contends with a problem that remains open for training more complex skills in the real world:

reward supervision. In this case, with previously proposed formalisms for learning from sparse human feedback. We believe that extending these works to learning from richer feedback, such as human preferences as described in Chapter 3 could be a fruitful direction for future work.

As we look toward realizing a future in which robots are capable of seamlessly integrating into daily life, there remain many open challenges. In the following, We will discuss some extensions of this work that would move us closer to this vision.

Towards general autonomous recovery A central argument of this thesis is that to enable autonomous robot systems, they must be able to learn on their own. The two fundamental requirements for this self-sufficiency in general are that robots must not rely on specific, environmental instrumentation or constant human supervision. In Chapter 2, we developed an agile recovery controller using simulation to enable a robot to autonomously recover without environment engineering or humans in the loop. There is preliminary evidence that this approach scales to more complex hardware such as humanoids, i.e., that we can transfer robust recovery controllers from simulation to enable greater autonomy for these legged robots. On the other hand, in manipulation applications, an approach to enable autonomous training is by defining a task graph, where the robot simultaneously learns tasks that reset each other [111, 322, 325, 326]. In both these cases, pre-training recovery controllers in simulation or pre-defining skills that reset each other, human involvement is required and this scaffolding limits the space of failures that a robot can recover from. Towards open-world deployment, a critical next step is tackling autonomous recovery from unanticipated failures. Chapter 8 describes a system that uses VLMs to perform reasoning in order to perform new, situationally-aware behaviors by guiding a language-conditioned manipulation policy. Such frameworks offer a promising foundation for enabling autonomous recovery in open-world situations.

Towards lifelong learning systems Much of this thesis described learning systems that learn "in real time". There has been this emphasis on efficiency because in real-world situations, it is important to adapt to the situation at hand in a reasonable amount of time as humans do when faced with new situations. However, humans also continuously get better over much longer time scales, i.e., over multiple hours, weeks, years, as a result of these learning experiences. Being able to *continually* learn is a critical piece of a practical robot system deployed in the world. In Chapter 5, we took a step in this direction; however, there are several axes that are underexplored. In particular, robots should not only be able to adapt a single skill to changing dynamics, but also continue to pick up multiple new skills, recall quickly how to perform in settings they have already learned in before, and become more efficient at learning over time.

Towards adapting generalist, robot foundation models Another limitation of the methods described in this thesis is that, for the most part, the robot is learning single tasks. Ultimately, generalist robots that can be deployed in open-world environments must have an architecture that supports flexible task specification and the ability to express a multitude of robot behaviors. Vision-language-action (VLA) models offer a promising architecture for this. We believe a fruitful future direction is leveraging VLAs as the base foundation and to explore how to imbue them with the ability to learn quickly from online, real-world experiences when deployed. VLAs are still nascent, though, so the most effective training mechanism for online learning remains unclear. However, due to their ability to perform reasoning, there are many possible avenues for exciting future work in this direction.

Chapter 10

Appendix

10.1 State Entropy Estimator

To approximate state entropy, we employ the simplified version of particle-based entropy estimator [144, 145]. Specifically, let $\mathbf{s}$ be a random variable with a probability density function p whose support is a set $\mathcal{S} \subset \mathbb{R}^q$. Then its differential entropy is given as $\mathcal{H}(\mathbf{s}) = -\mathbb{E}_{\mathbf{s}\sim p(\mathbf{s})}[\log p(\mathbf{s})]$. When the distribution p is not available, this quantity can be estimated given N i.i.d realizations of $\{\mathbf{s}_i\}_{i=1}^N$ [144]. However, since it is difficult to estimate p with high-dimensional data, particle-based k-nearest neighbors (k-NN) entropy estimator [145] can be employed:

$$\widehat{\mathcal{H}}(\mathbf{s}) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{N \cdot ||\mathbf{s}_i - \mathbf{s}_i^k||_2^q \cdot \widehat{\pi}^{\frac{q}{2}}}{k \cdot \Gamma(\frac{q}{2}+1)} + C_k \tag{10.1}$$

$$\propto \frac{1}{N}\sum_{i=1}^{N} \log ||\mathbf{s}_i - \mathbf{s}_i^k||_2, \tag{10.2}$$

where $\widehat{\pi}$ is the ratio of a circle's circumference to its diameter, $\mathbf{s}_i^k$ is the k-NN of $\mathbf{s}_i$ within a set $\{\mathbf{s}_i\}_{i=1}^N$, $C_k = \log k - \Psi(k)$ a bias correction term, Ψ the digamma function, Γ the gamma function, q the dimension of $\mathbf{s}$, and the transition from (10.1) to (10.2) always holds for $q > 0$. Then, from Equation 10.2, we define the intrinsic reward of the current state $\mathbf{s}_t$ as follows:

$$r^{\texttt{int}}(\mathbf{s}_t) := \log\left(\left\|\mathbf{s}_t - \mathbf{s}_t^k\right\|\right)$$

10.2 Experimental Details

Training details . For our method, we use the publicly released implementation repository of the SAC algorithm (`https://github.com/denisyarats/pytorch_sac`) with a

Hyperparameter	Value	Hyperparameter	Value
Initial temperature	0.1	Hidden units per each layer	1024 (DMControl), 256 (Meta-world)
Learning rate	0.0003 (Meta-world), 0.001 (cheetah)	Batch Size	1024 (DMControl), 512 (Meta-world)
	0.0001 (qauadruped), 0.0005 (walker)	Optimizer	Adam [75]
Critic target update freq	2	Critic EMA τ	0.005
(β_1, β_2)	$(.9, .999)$	Discount γ	.99

Table 10.1: Hyperparameters of the SAC algorithm. Most hyperparameters values are unchanged across environments with the exception for learning rate.

Hyperparameter	Value	Hyperparameter	Value
GAE parameter λ	0.92	Hidden units per each layer	1024 (DMControl), 256 (Meta-world)
Learning rate	0.0003 (Meta-world), 0.0001 (quadruped)	Batch Size	512 (cheetah), 128 (Otherwise)
	$5e^{-5}$ (quadruped, Walker)	# of timesteprs per rollout	100 (cheetah, Walker), 500 (quadruped)
# of environments per worker	16 (quadruped, cheetah), 32 (Walker)	PPO clip range	0.2
Entropy bonus	0.0	Discount γ	.99

Table 10.2: Hyperparameters of the PPO algorithm. Most hyperparameters values are unchanged across environments with the exception for learning rate.

full list of hyperparameters in Table 10.1. On the DMControl environments, we use segments of length 50 and a frequency of teacher feedback of 20K timesteps, which corresponds to roughly 20 episodes. We choose the number of queries per feedback session $M = 140, 70, 40$ for the maximum budget of 1400, 700, 400 on Walker and Cheetah, and choose $M = 70, 35, 20$ for the maximum budget of 1400, 700, 400 on Quadruped. For Meta-world, we use segments of length 10 and set $M = 64, K = 2400$ for the maximum budget of 2500, 5000, and 10000 on Drawer Close, Window Open, Door Open, and Button Press and $M = 128, K = 4800$ for maximum budget of 25000, 50000 on Sweep Into and Drawer Open.

For preference PPO, we use the publicly released implementation repository of the PPO algorithm (`https://github.com/DLR-RM/stable-baselines3`) with a full list of hyperparameters in Table 10.2. We choose the number of queries per feedback session $M = 70, 45$ for the maximum budget of 2100, 1400 on the DMControl environments. For the reward model, we use same setups for our method. For Meta-world, we use segments of length 10 and set $M = 256, K = 2400$ for all environments and budgets of feedback.

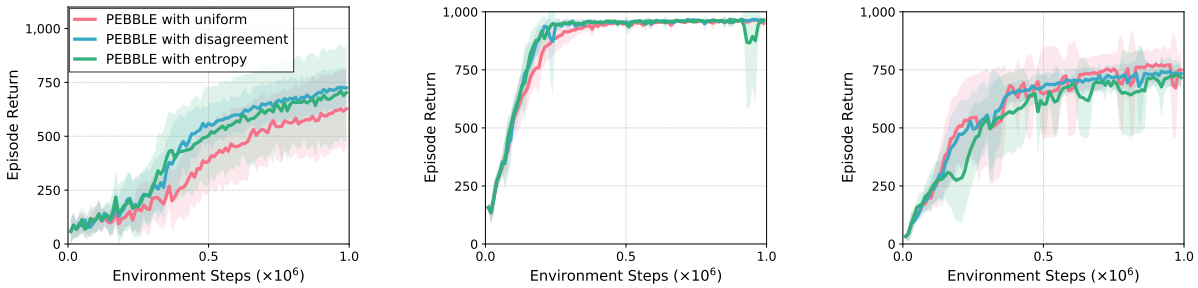
Reward model For the reward model, we use a three-layer neural network with 256 hidden units each, using leaky ReLUs. To improve the stability in reward learning, we use an ensemble of three reward models, and bound the output using tanh function. Each model is trained by optimizing the cross-entropy loss defined in (3.4) using ADAM

learning rule [75] with the initial learning rate of 0.0003.

Environments We follow the standard evaluation protocol for the benchmark locomotion tasks from DMControl. The Meta-world single-task benchmark involves training and testing on a single instantiation (fixed reset and goal) of the task. To constitute a more realistic single-task manipulation setting, we randomize the reset and goal positions in all our experiments. We also use new reward function, which are nicely normalized and make the tasks stable.

10.3 Effects of Sampling Schemes

Figures 10.1 and 10.2 show the learning curves of PEBBLE with various sampling schemes. For Quadruped, we find that the uncertainty-based sampling schemes (using ensemble disagreement or entropy) are superior to the naive uniform sampling scheme. However, they did not lead to extra gains on relatively simple environments, like Walker and Cheetah, similar to observations from Ibarz et al. [125]. Similarly, on the robotic manipulation tasks, we find little difference in performance for simpler tasks (Drawer Close, Window Open). However, we find that the uncertainty-based sampling schemes generally fare better on the other environments.



(a) Quadruped (b) Walker (c) Cheetah

Figure 10.1: **(Locomotion sampling ablations)** Learning curves of PEBBLE with 1400 pieces of feedback by varying sampling schemes. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs.

10.4 Examples of Selected Queries

Figure 10.3, 10.4 and 10.5 show some examples from the selected queries to teach the agents.

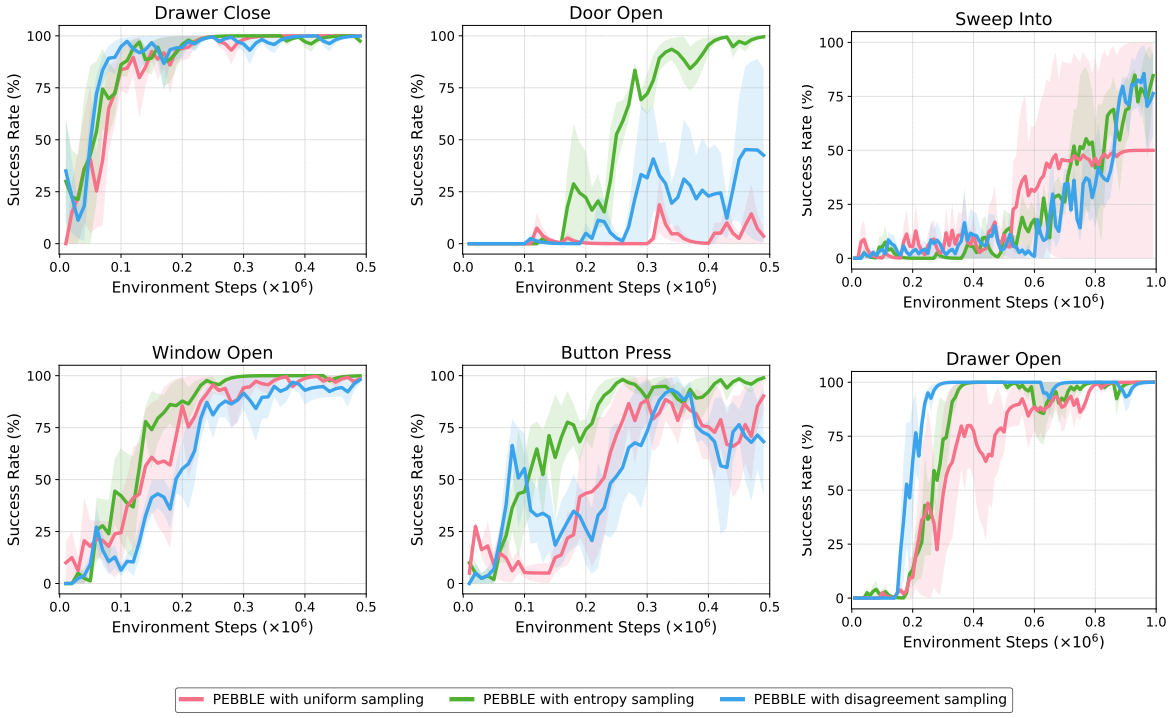
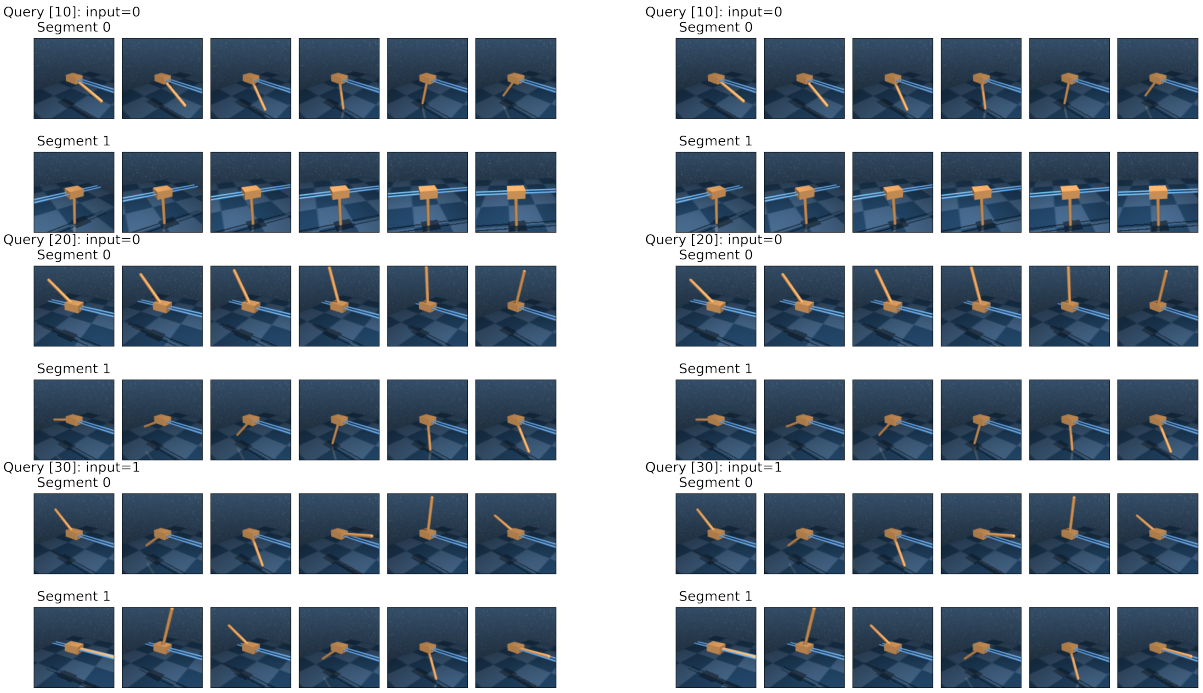


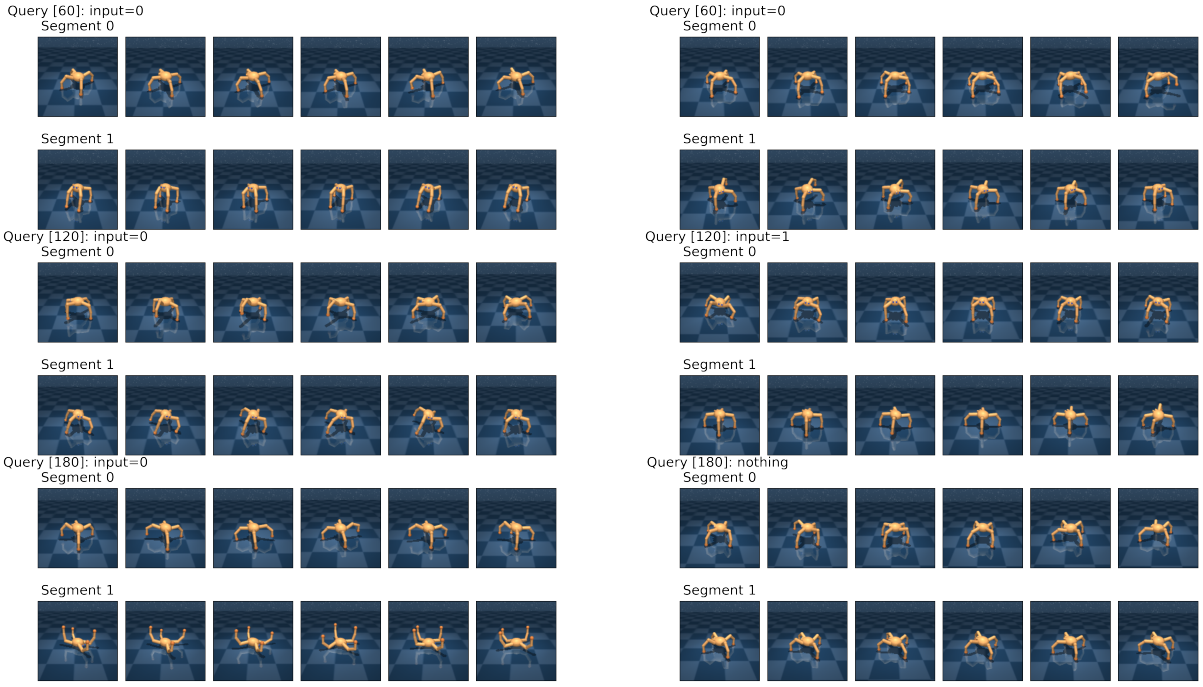
Figure 10.2: **(Meta-world sampling ablations)** Learning curves of PEBBLE with various sampling schemes on the Meta-world tasks. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten runs.



(a) Clock-wise windmill

(b) Counter clock-wise windmill

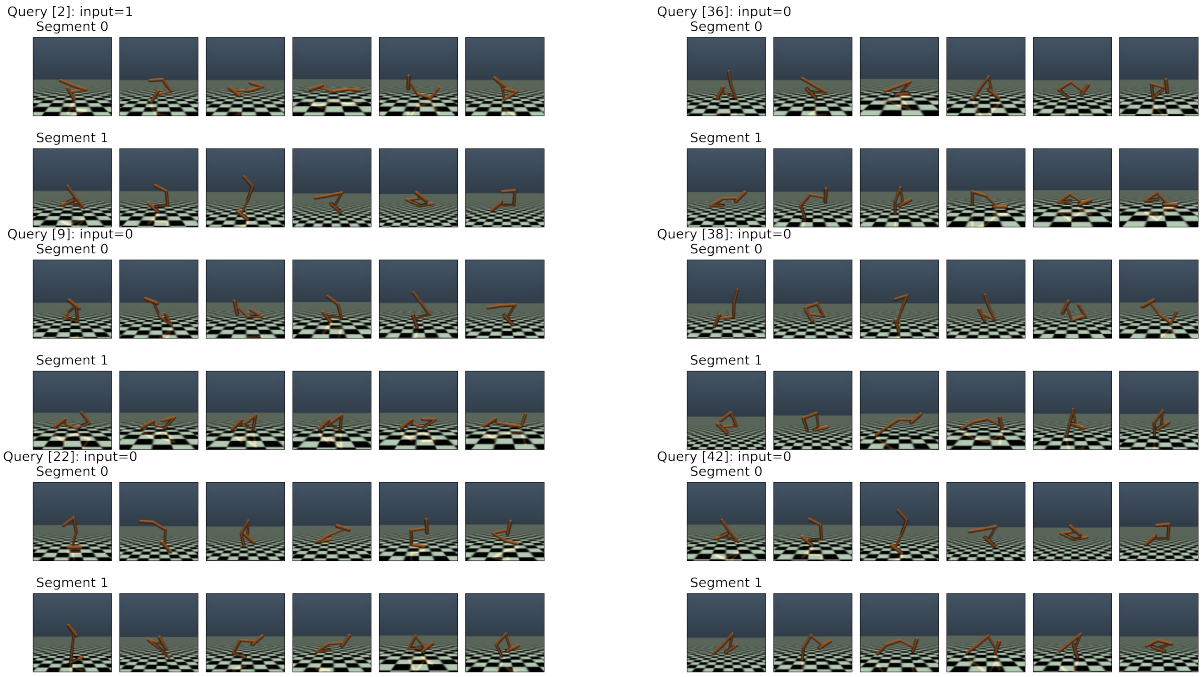
Figure 10.3: **(Examples from the selected queries to teach the Cart agent)**



(a) Waving left front leg

(b) Waving right front leg

Figure 10.4: **(Examples from the selected queries to teach the Quadruped agent)**



(a) Query example 1

(b) Query example 2

Figure 10.5: **(Examples from the selected queries to teach the Hopper agent)**

10.5 Detailed Experiments

10.5.1 Sparse Adroit

Full Results Here we present the full results for Sparse Adroit, including LayerNorm and environment-specific design choice ablations.

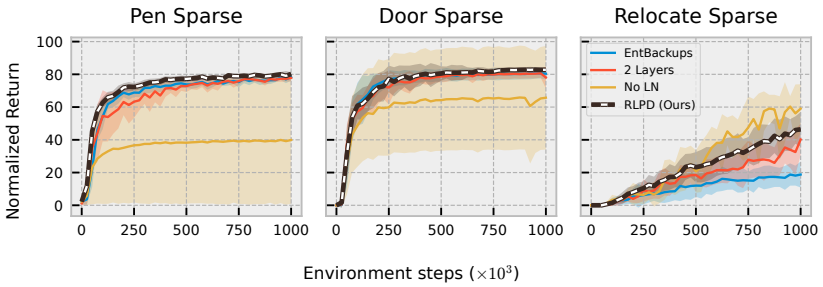


Figure 10.6: **(Full Adroit results)**

We see that LayerNorm greatly reduces variance on the Pen and Door environments, however slightly harms mean performance on Relocate. Taken together however, LayerNorm is still a vital ingredient for reliable performance on this domain.

10.5.2 AntMaze

Full Results Here we present the full results for AntMaze, including LayerNorm and environment-specific design choice ablations.

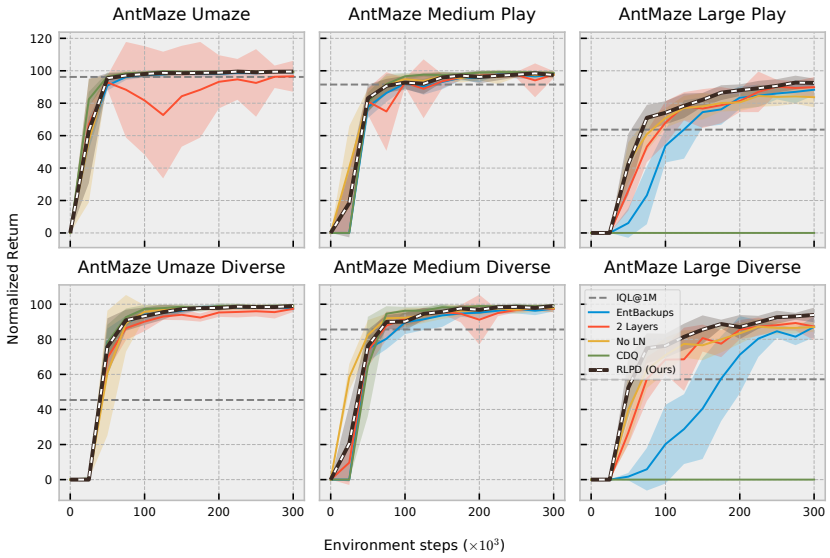


Figure 10.7: **(Full AntMaze results)**

We see that our recommended design choices are vital for strong performance on the harder Large tasks, and furthermore see that using deeper 3-layer networks seems to help stability across all tasks. As we see, the best design choices on the Large environments are also optimal for the Umaze and Medium environments.

Gradient Steps per Environment Step Ablations We see LayerNorm is incredibly effective in a setting whereby we perform a single gradient update per time-step in Figure 10.8. This may be required for learning on real robots, where computational efficiency is important.

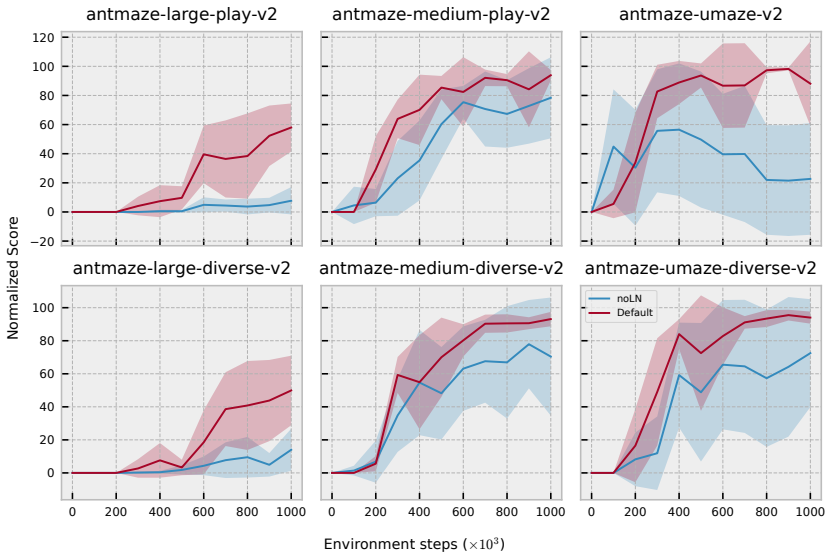


Figure 10.8: **(SAC with and without Layer Normalization)**

In Figure 10.9, we see the impact of increasing the number of gradient steps per time-step. As we see, both sample efficiency stability improves greatly.

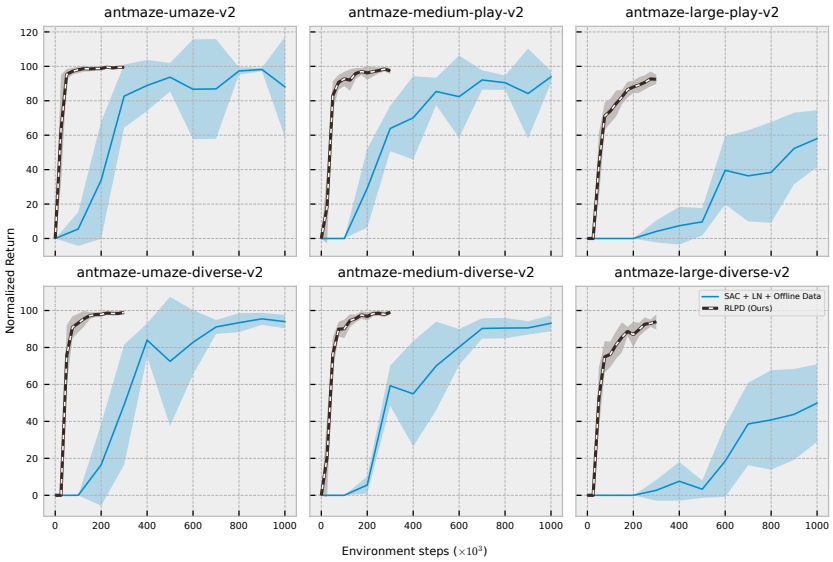


Figure 10.9: **(RLPD v.s. SAC + LN + Offline Data)**

Full IQL comparison Here we compare to IQL + Finetuning. We show pre-training gradient steps on IQL as negative indices in the x-axis. As we see in Figure 10.10, despite IQL learning a strong initialization, it struggles to improve beyond this. In comparison, RLPD is able to quickly match, then greatly exceed IQL.

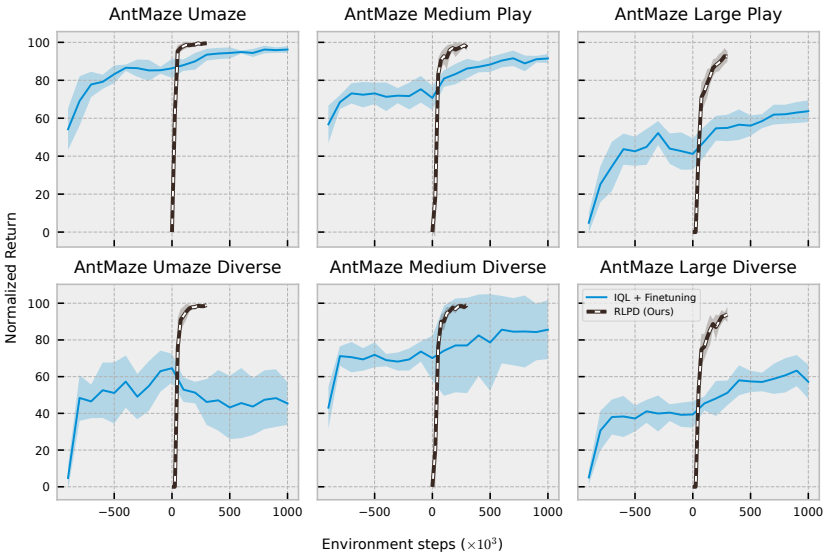


Figure 10.10: **(RLPD v.s. IQL + Finetuning)** We show offline pre-training steps for IQL using negative indices.

10.5.3 D4RL Locomotion

Full Results Here we present the full results for D4RL Locomotion, including LayerNorm and baselines. We emphasize that these tasks are not necessarily a good use case for online learning with offline datasets, as the tasks can be solved relatively quickly with purely online approaches, and there is no inherent difficulty regarding exploration.

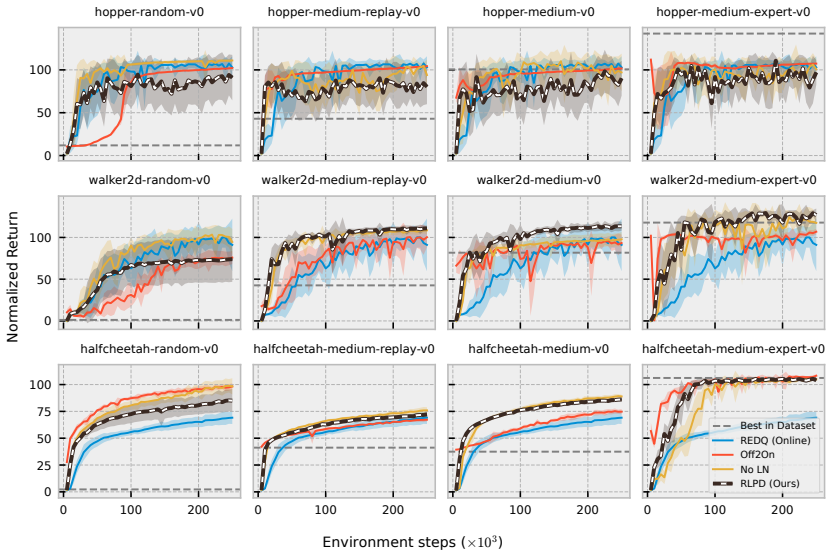


Figure 10.11: **(D4RL Ablations)**

The impact of LayerNorm is not so clear cut in Figure 10.11; this is to be expected as online approaches already achieve strong results in this domain. Notably, we see strong performance compared to baselines in the medium-expert domains.

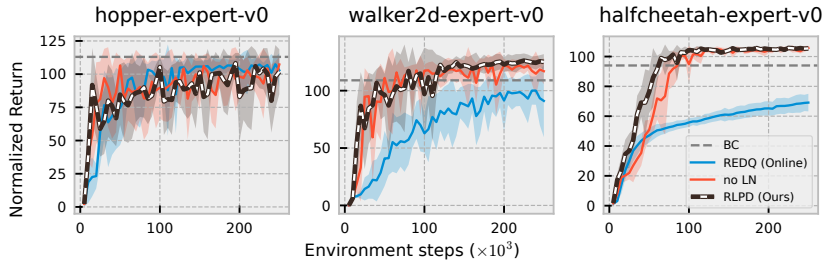


Figure 10.12: **(D4RL Ablation on Expert Data)**

We also test on the narrow Expert dataset, not tested by [215]. In Figure 10.12 we see LayerNorm can marginally help both sample efficiency, and asymptotic performance.

10.5.4 V-D4RL Locomotion

Full Results Here we present the full results for V-D4RL Locomotion, including LayerNorm and environment-specific design choice ablations.

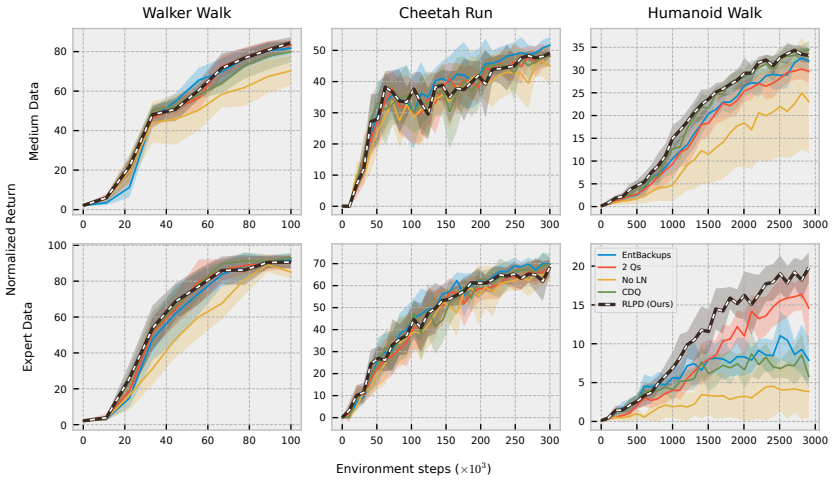
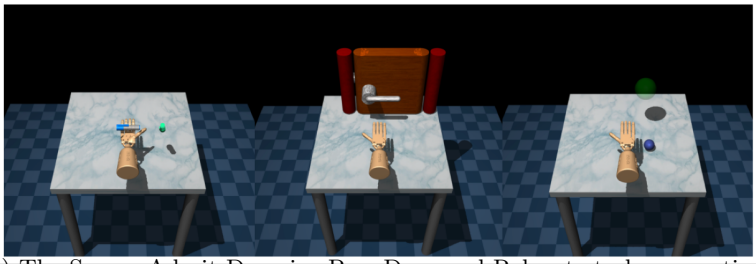


Figure 10.13: **(VD4RL Ablations)**

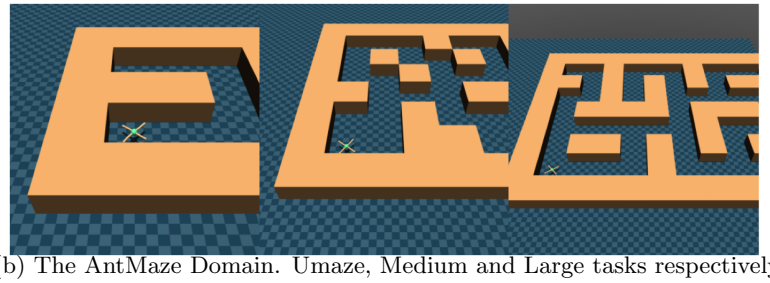
As we see, LayerNorm helps significantly in the Walker and Humanoid environments. We also see the positive impact of our recommended design choices in the complex Humanoid domain.

10.6 Experimental Details

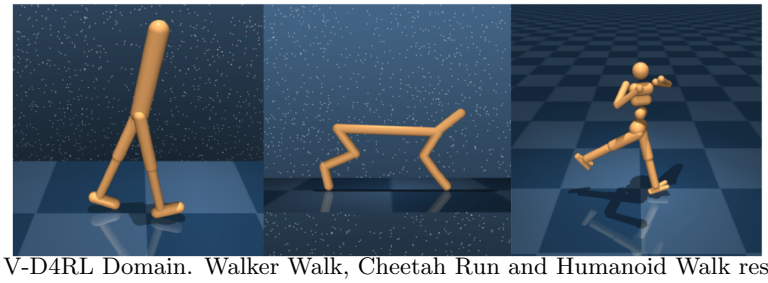
10.6.1 Further Environment Details



(a) The Sparse Adroit Domain. Pen, Door and Relocate tasks respectively.



(b) The AntMaze Domain. Umaze, Medium and Large tasks respectively.



(c) The V-D4RL Domain. Walker Walk, Cheetah Run and Humanoid Walk respectively.

Figure 10.14: **(Experimental environment visualization)**

We provide further details about the key domains we evaluate on. In Figure 10.14 we provide visualizations of the environments.

Sparse Adroit In these tasks, reward is a binary variable that indicates whether the task has been completed successfully or not. In prior work [210, 221], it is common to see success rate used as the metric, which simply determines if the task has been completed in any time step. However Kostrikov et al. [216] use a more challenging metric that involves speed of completion. Concretely, return is calculated as the percentage of the total timesteps in which the task is considered solved (note that there are no early terminations). For example, in the 'Pen' task, where the horizon is 100 timesteps, if a

policy achieves a Normalized Score of 80, that means in 80 timesteps, the task is considered solved. This effectively means that the policy was able to solve the task in 20 timesteps. At each evaluation, we perform 100 trials.

D4RL: AntMaze In these tasks, reward is a binary variable that indicates whether the agent has reached the goal. Upon reaching the goal, the episode terminates. The normalized return is therefore the proportion of evaluation trials that were successful. We follow prior work, and perform 100 trials, and measure Normalized Return as the percentage of successful trials.

10.6.2 Hyperparameters

Here we list the hyperparameters for RLPD in Table 10.3, and the environment-specific hyperparameters in Table 10.4.

Table 10.3: RLPD hyperparameters.

Parameter	Value
Online batch size	128
Offline batch size	128
Discount (γ)	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Ensemble size (E)	10
Critic EMA weight (ρ)	0.005
Gradient Steps (State Based) (G or UTD)	20
Network Width	256 Units
Initial Entropy Temperature (α)	1.0
Target Entropy	$-\dim(\mathcal{A})/2$
Pixel-Based Hyperparameters	
Action repeat	2
Observation size	[64, 64]
Image shift amount	4

Table 10.4: Environment specific hyperparameters.

Environment	CDQ	Entropy Backups	MLP Architecture
Locomotion	True	True	2 Layer
AntMaze	False	False	3 Layer
Adroit	True	False	3 Layer
DMC (Pixels)	False	False	2 Layer

10.6.3 Experimental Details

10.6.3.1 Further Environment Details

Observations We define the state $\mathbf{s}_t$ to be the 2D Cartesian heading to the goal in the global frame, plus a three-step history of the following features: root roll and pitch, roll and pitch velocity, joint angles, root displacement, and previous actions. Additionally, the bipedal navigation policy receives 3D vectors representing its root x and z axes in the global frame. Roll and pitch, root displacement, goal heading, and the axis orientations come from a motion capture system. Root displacement is a 3D Cartesian vector, the position of the robot's torso relative to its position at the beginning of the episode. Roll and pitch velocity come from the robot's internal IMU. For the hurdling policy, the goal in the goal heading is the middle of the next hurdle. When the robot's root position is 40cm past one hurdle, the goal heading begins pointing to the next hurdle.

Actions The actions are position targets for the 12 joints of the robot. The policy outputs offset o for each joint from a nominal pose, which for each leg is $p = [0.0, 0.9, -1.8]$, at a frequency of 20Hz. We define a range for the offsets so as to obey joint limits as follows: $o_{min} = [-0.6, -1.2, -1.2], o_{max} = [0.6, 1.2, 1.2]$, for each leg. We use a position controller to compute the torques to be applied $\tau = K_p(q^* - q) - K_d \cdot \dot{q}$, where q^* and q are the desired and current joint positions, respectively, and $\dot{q}$ represents the joint velocities. K_p and K_d are the gains and damping for the motors. We find the values for K_p and K_d to be important in learning our tasks, and set them to $K_p = 60$ and $K_d = 3.0$. We smooth the actions from the policy using a low-pass Butterworth filter with frequency cutoff $= 4.0$.

Jumping Task We give all details necessary to reproduce our jumping results. First, we detail how we obtained the source policy, then how our task is implemented.

Training the source policy. The jumping source policy is trained to imitate a reference motion clip using the RL framework proposed by [57]. Given a reference motion $\mathcal{M}$ comprising a sequence of poses, the policy is trained to imitate the motion using a reward function that encourages tracking the target poses at each timestep (see Table 10.6). The state $\mathbf{s}_t$ contains a history of 3 timesteps for each of the following features: root orientation, joint angles, and previous actions. The policy also receives a goal $\mathbf{g}_t$, which comprises the target poses (root position, root rotation, and joint angles) calculated from the reference motion for future timesteps. In our experiments, we use 4 future target poses, the latest of which is a target for approximately 1 second ahead of the current timestep. We adopt the reward function from [57], where the reward r_t at each timestep is calculated according to:

$$r_t = \sum_i w^i r_t^i \tag{10.3}$$

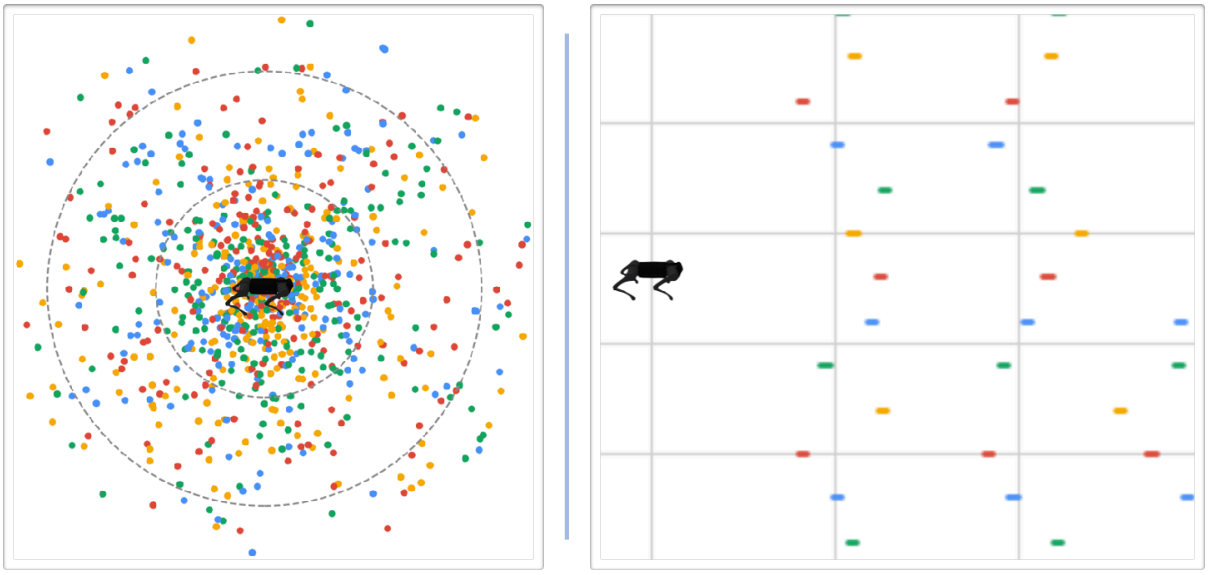


Figure 10.15: **(Distribution of training goals)** (Left) We visualize the bipedal navigation task, where each colored dot is a sample from the goal distribution we defined. The small dotted circle corresponds to a radius of 1m and the larger circle corresponds to a radius of 2m.

with w's and r's listed in Table 10.5.

Reward	Formula	Weight	Explanation
Joint pose	$\exp\left[-5\sum_j \|\|q^j_{\text{ref}} - q^j_{\text{robot}}\|\|^2\right]$	0.5	q^j = 1D local rotation of joint j
Joint velocity	$\exp\left[-0.1\sum_j \|\|\dot{q}^j_{\text{ref}} - \dot{q}^j_{\text{robot}}\|\|^2\right]$	0.05	
Root pose	$\exp\left[-20\|\|\mathbf{x}^{\text{root}}_{\text{ref}} - \mathbf{x}^{\text{root}}_{\text{robot}}\|\|^2 - 10\|\|\mathbf{q}^{\text{root}}_{\text{ref}} - \mathbf{q}^{\text{root}}_{\text{robot}}\|\|^2\right]$	0.15	$\mathbf{x}^{\text{root}}$ = root's global position, $\mathbf{q}^{\text{root}}$ = root's rotation
Root velocity	$\exp\left[-2\|\|\dot{\mathbf{x}}^{\text{root}}_{\text{ref}} - \dot{\mathbf{x}}^{\text{root}}_{\text{robot}}\|\|^2 - 0.2\|\|\dot{\mathbf{q}}^{\text{root}}_{\text{ref}} - \dot{\mathbf{q}}^{\text{root}}_{\text{robot}}\|\|^2\right]$	0.1	
End effector	$\exp\left[-40\sum_e \|\|\mathbf{x}^e_{\text{ref}} - \mathbf{x}^e_{\text{robot}}\|\|^2\right]$	0.2	$\mathbf{x}^e$ = robot-relative 3D position of end effector e

Table 10.5: Reward terms encouraging tracking a jumping reference motion, from [57].

Our task. Jump consecutively over 20cm wide, 1cm tall 'hurdles' that are spaced uniformly at random between 1.6 and 2.6 meters apart (see Figure 10.15). As such, the robot should run up to each hurdle to jump over it, then land while continuing to run in order to execute the next jump at the right time. We design our reward function with minimal shaping, so as to ensure that it is amenable to adapt to other environments. To ensure that it jumps over the hurdles, we design the reward function to be non-negative and terminate the episode on contact with the hurdle. The reward r_t at each timestep is calculated according to Equation 10.3 with w's and r's listed in Table 10.6.

Reward	Formula	Weight	Explanation
Forward	1 if $v_x > 0.5$ else 0.007	0.7	Move toward the next hurdle
Orientation	$\exp\left[-2\|\|\mathbf{q}_{\text{root}}\|\|\right]$	0.05	Stay upright and pointing forward
Joint velocity	$\exp\left[-0.1\|\|\dot{\mathbf{q}}_{\text{joints}}\|\|\right]$	0.15	Use smoother motions
Hurdle	count [hurdles passed]	0.1	Bonus for clearing each hurdle

Table 10.6: Exact definition of each of the reward terms used in our overall jumping reward function defined in Equation 10.3.

Bipedal Navigation Task Now we give all details necessary to reproduce our bipedal navigation policy.

Training the source policy. We define a goal-conditioned navigation task in which the robot must acquire the agility to get up on its hind legs and walk to a desired location while maintaining balance. As such, we first train a policy *just* to get up onto its hind legs by training from scratch with a reward function that sums two terms: (i) r^{upright}, a term to make the body perpendicular to the ground and (ii) $r^{\text{max height}}$ a term to maximize the height of the robot's CoM. For r^{upright}, we first get the robot's forward vector $\mathbf{v}_{\text{forward}}$ by rotating $[1, 0, 0]$ by the robot's orientation. We then compute the cosine distance

$$\texttt{cos\_dist} = \mathbf{v}_{\text{forward}}^{\top}[0, 0, 1].$$

Lastly, we normalize to be between 0 and 1:

$$r^{\text{upright}} = (0.5 \cdot \texttt{cos\_dist} + 0.5)^2. \tag{10.4}$$

To encourage the robot to lift itself up, we define

$$r^{\text{max height}} = \exp(\texttt{root\_height}) - 1, \tag{10.5}$$

so a height of 0 would correspond to 0 reward and increase with the height exponentially thereafter. The final reward is

$$r^{\text{stand}} = r^{\text{upright}} + r^{\text{max height}} \tag{10.6}$$

We terminate the episode if any part of the robot other than its feet touch the ground.

Our task. We sample goals uniformly at random along the perimeter of a circle, whose radius is sampled from an exponential distribution with mean 3/2, so as to train the robot to walk to a wide variety of goal locations (see left of Figure 10.15). The reward

function is a combination of the standing reward and a simple forward locomotion reward. Specifically, we define the reward to be

$$r_t^{\text{stand}} \cdot (1 + r_t^{\text{facing}} + r_t^{\text{distance}}). \tag{10.7}$$

where r_t^{facing} follows Equation 10.4, where `cos_dist` is computed between the robot's down vector and its global displacement to the goal. Intuitively, this encourages the robot's belly to point directly at the goal (or for it to face the goal). Finally, the distance reward encourages progress towards the goal:

$$r_t^{\text{distance}} = \begin{cases} 1, & \text{if } \texttt{curr\_distance} \leq .01 \\ .5, & \text{if } \texttt{curr\_distance} < \texttt{prev\_distance} \\ 0, & \text{otherwise.} \end{cases}$$

where `curr_distance` is the l_2 distance from the robot to the goal, and `prev_distance` is that from the previous timestep.

Transfer to New Environments The target sim environments that we construct to study shifts in dynamics are:

(i) *Bumpy:* We generate a random heightfield with a maximum height of 5 centimeters. The robot is reset in random positions on the terrain.

(ii) *Icy:* We modify the lateral friction coefficient to 0.4.

(iii) *Weakened:* We attempt to simulate real-world stochastic motor performance dynamics by randomly weakening legs and shifting motor temperatures.

(iv) *Sloped:* We place the robot next to and point it towards an 8-degree incline.

(v) *Gravity:* We simulate low-gravity environments by modifying the gravity to 3.7 m/s^2. We found that the gravity on the moon (1.62 m/s^2) was not amenable to the jumping task, so we raised it so as to allow the robot to eventually settle.

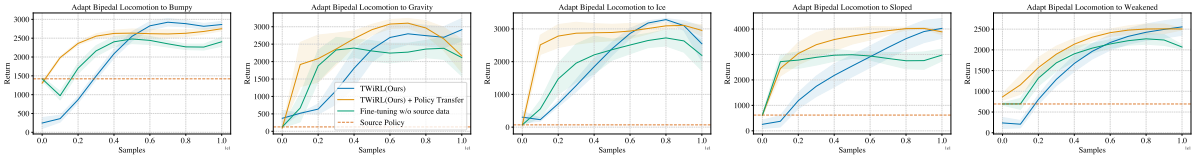
Since for some of the domains, the task only makes sense for certain goals, we do not randomly sample goals for the dynamics fine-tuning task. For example, to contend with the dynamics shift imposed by the incline, the goal needs to be placed *on* the hill.

10.6.3.2 Training

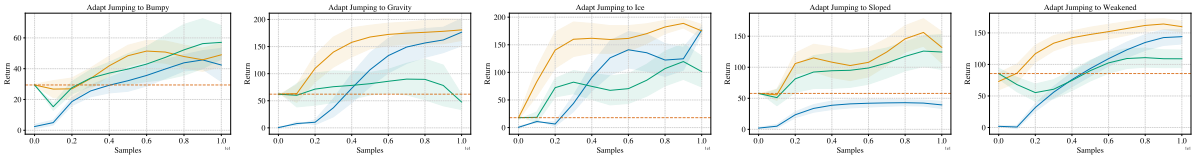
We use the open-sourced JAX implementation of the DroQ algorithm (`https://github.com/ikostrikov/walk_in_the_park`) with a full list of hyperparameters in Table 10.7.

Parameter	Value
Batch size	256
Discount (γ)	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Critic EMA weight (ρ)	0.005
UTD ratio	20
Network Width	256 Units
Network Depth	2 Layers
Initial Entropy Temperature (α)	1.0
Target Entropy	$-\dim(\mathcal{A})/2$
TWiRL Hyperparameters	
Source sampling ratio (ϕ)	0.5

Table 10.7: Hyperparameters shared for training source policies with DroQ and for transferring with Algorithm 7.



(a) Adapting the bipedal navigation policy to the 5 target environments.



(b) Adapting the jumping policy to the 5 target environments.

Figure 10.16: **(Full dynamics adaptation results)** Individual learning curves for every task and target environment combination.

10.6.4 Detailed Experiments

10.6.4.1 Full results

We present all learning curves for evaluating TWiRL's ability to facilitate transfer to different environments in Figure 10.16. We see that in every case except for adapting the jumping policy to the sloped environment, vanilla TWiRL successfully learns a policy that works in the new environment. Again, we emphasize that vanilla TWiRL does *not* start with a pre-trained policy, we simply start with the replay from π_{src}. We see that by transferring the weights works robustly in every case. Lastly, we also emphasize that incorporating the data from the source policy is essential, as we see that training from the pre-trained policy with only data collected in the new environment trains significantly slower than either variant of our method.

10.6.4.2 Ablations

In this section, we conduct a series of ablation studies to evaluate how different design decisions affect the performance of our method. Our focus is on two axes, (i) how much does incorporating *data* from the source policy during training affect learning and (ii) whether using a sample efficient, or using a high *update-to-data* ratio, algorithm to efficiently incorporate the data matters. For these experiments, we use dynamics transfer as our goal with a budget of 100k samples for fine-tuning as we can then test *multiple* target environments.

Data ablation Our method updates the policy π_θ using data from both the offline source buffer $\mathcal{D}_{\text{src}}$ and an online replay buffer $\mathcal{D}_{\text{target}}$ with a fixed ratio ϕ, with the pro-

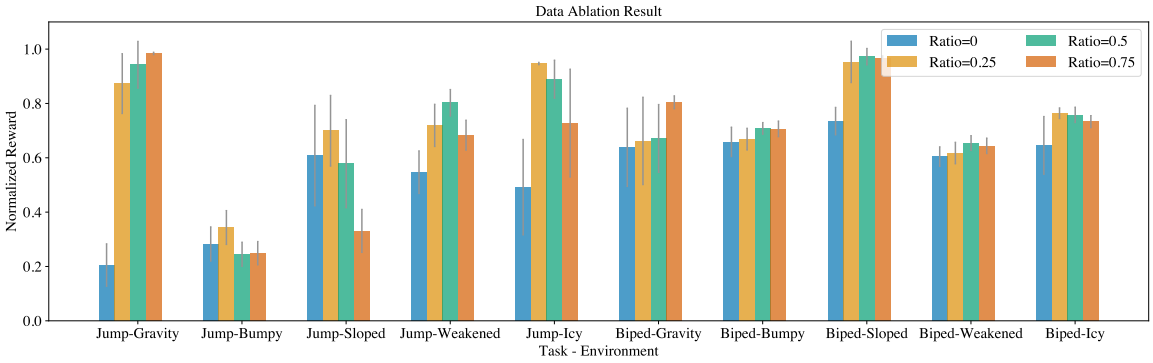


Figure 10.17: **(Effect of using data from $\pi_{\mathbf{src}}$)**

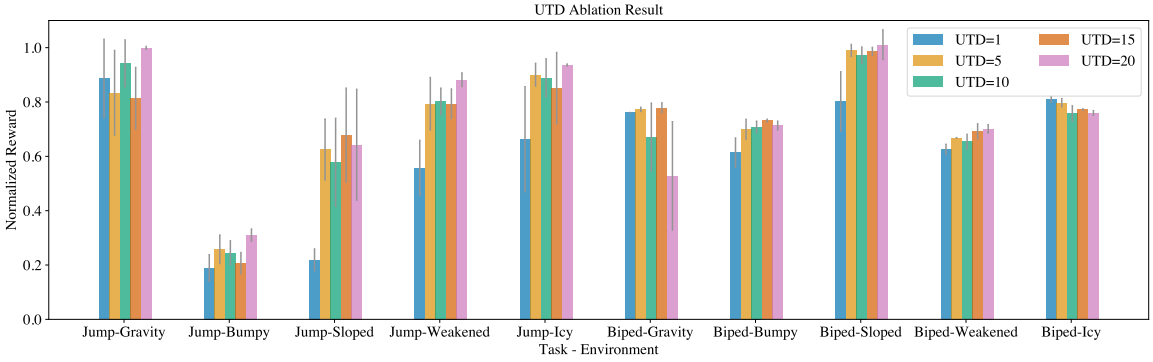


Figure 10.18: **(Effects of using high UTD algorithms for incorporating off-policy data)** We find that using an UTD ratio > 1 is important for stable learning particularly when learning the jumping policies in a new environment.

cedure defined in Algorithm 7. Here we test 4 ratio options: [0, 0.25, 0.5, 0.75], where 0 indicates that we *only sample data from the online replay buffer*, meaning we do not incorporate the source policy data at all. $\phi = 0.5$, which we end up using in our main experiments, indicates we sample half the data from the offline buffer and the other half from the online buffer. Results are shown in 10.17. Intuitively, discarding the pre-training data and sampling entirely from the online experience (corresponding to $\phi = 0$) should allow the agent to most accurately fit to the test environment, as its models do not need to fit to heterogeneous data. However, at the start of fine-tuning as the model needs to adapt to data from a different distribution with such little data, we observe this to be very unstable. By incorporate the source buffer, the off-policy RL algorithm exploits the inherent similarities between the source and target environment to improve the policy update. The result shows that higher usage of data from source buffer does not lead to

higher result since the lack of data from current environment can prohibit the algorithm to understand the property of the target environment, $\phi = 0.5$ result with the best performance in most cases. The result also shows that the improvement by incorporating pre-training dataset differs between target environments, some can greatly improve the fine-tuning outcome e.g. icy, weakened, while the progress is limited for some environments. Our assumption is that the difference between the source and target environment can affects the result. We leave the investigation of how to measure the similarity between environment to future work.

Bibliography

[1] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf. 1

[2] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 739–746, 2005. URL https://proceedings.neurips.cc/paper/2005/hash/fdf1bc5669e8ff5ba45d02fded729feb-Abstract.html.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

[4] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9328–9337. IEEE, 2019. doi: 10.1109/ICCV.2019.00942. URL https://doi.org/10.1109/ICCV.2019.00942. 1

[5] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *International Conference on Robotics and Automation*, 2018. 1, 21

[6] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, et al. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8kbp23tSGYv. 103, 104

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025. 106, 109, 111

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023. 103

[9] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems (RSS)*, 2023. 103

[10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, et al. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024. 103, 104, 106

[11] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2024. URL https://proceedings.mlr.press/v270/kim25c.html. 1, 103, 110, 111

[12] Joshua Tobin, Rachel Fong, Alex Ray, J. Schneider, W. Zaremba, et al. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. 2, 37, 87

[13] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, et al. Solving rubik's cube with a robot hand. *ArXiv*, abs/1910.07113, 2019. 37

[14] X. Peng, Marcin Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018. 9, 37, 38, 87

[15] Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Sci. Robotics*, 4(26), 2019. doi: 10.1126/SCIROBOTICS.AAU5872. URL https://doi.org/10.1126/scirobotics.aau5872. 87

[16] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[17] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher G. Atkeson, Sören Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pages 73–92. PMLR, 2023. URL https://proceedings.mlr.press/v229/zhuang23a.html.

[18] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, page 02783649241285161, 2024.

[19] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model. *arXiv preprint arXiv:2411.14386*, 2024.

[20] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024. 2

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236. 2, 66

[22] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, January 2017. doi: 10.1038/nature16961. 21, 66

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Artfhur Guez, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[24] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michael Mathieu, Andrew Dudzik, Junyoung Chung, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 2, 21

[25] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018. 2, 23

[26] Anusha Nagabandi, K. Konolige, Sergey Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. *Conference on Robot Learning (CoRL)*, abs/1909.11652, 2019. 50

[27] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 1110–1120. PMLR, 2020. URL https://proceedings.mlr.press/v155/ha21c.html. 2, 10, 14, 38, 39, 40, 42, 47, 53, 62

[28] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In Hadas Kress-Gazit, Siddhartha S. Srinivasa, Tom Howard, and Nikolay Atanasov, editors, *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. doi: 10.15607/RSS.2018.XIV.010. URL http://www.roboticsproceedings.org/rss14/p10.html. 2, 7, 9, 11, 38, 52, 86

[29] Wenhao Yu, V. Kumar, Greg Turk, and C. Liu. Sim-to-real transfer for biped locomotion. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510, 2019. 9, 16, 37, 38, 87

[30] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, et al. Policies modulating trajectory generators. *Conference on Robot Learning (CoRL)*, abs/1910.02812, 2018. 38, 39, 91

[31] Jemin Hwangbo, J. Lee, A. Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, et al. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4, 2019. 7, 9, 37, 38, 86

[32] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5, 2020. 7, 9, 11, 38, 40, 42, 43, 84, 91

[33] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 2021. URL `https://proceedings.mlr.press/v164/rudin22a.html`. 37, 38, 40, 87

[34] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems (RSS)*, 2021. 2, 8, 9, 16, 37, 38, 42, 52, 54, 56, 86

[35] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017. 3, 22, 23, 25, 27, 29, 30, 31

[36] Laura Smith, J. Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, et al. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. 4, 38, 39, 42, 43, 44, 47

[37] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning (ICML)*, 2021. 5

[38] Laura Smith, Ilya Kostrikov, and Sergey Levine. Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *Robotics: Science and Systems (RSS) Demo*, abs/2208.07860, 2023. 5, 53, 54, 56, 58, 62, 91

[39] Laura Smith, Yunhao Cao, and Sergey Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion. *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. 5

[40] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023. 5

[41] Laura Smith, J. Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, et al. Learning and adapting agile locomotion skills by transferring experience. *Robotics: Science and Systems (RSS)*, 2024. 5, 52, 56

[42] Laura Smith, Alex Irpan, Montserrat Gonzalez Arenas, Sean Kirmani, Dmitry Kalashnikov, Dhruv Shah, and Ted Xiao. Steer: Flexible robotic manipulation via dense language grounding. *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. 5

[43] Ian D. Miller, A. Cohen, Adarsh Kulkarni, James D. Laney, C. J. Taylor, et al. Mine tunnel exploration using multiple quadrupedal robots. *IEEE Robotics and Automation Letters*, 5:2840–2847, 2020. 7

[44] G. Bledt, Matthew J. Powell, B. Katz, J. Carlo, P. Wensing, et al. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018. 7, 38, 52, 86

[45] Christian Gehring, Stelian Coros, M. Hutter, Dario Bellicoso, Huub Heijnen, et al. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics and Automation Magazine*, 23:34–43, 2016.

[46] M. Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, Dario Bellicoso, et al. Anymal - a highly mobile and dynamic quadrupedal robot. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016. 7, 9, 38, 52, 86

[47] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 3:2619–2624 Vol.3, 2004. 7, 9, 37, 39, 50, 52, 86, 91

[48] Russ Tedrake, T. Zhang, and H. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3:2849–2854 vol.3, 2004. 9, 37, 39, 52, 86

[49] Russ Tedrake and H. Seung. Learning to walk in 20 minutes. 2005. 39

[50] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning CPG sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 1267–1273. AAAI Press / The MIT Press, 2005. URL http://www.aaai.org/Library/AAAI/2005/aaai05-201.php. 9, 37, 39, 52, 86

[51] N. Heess, TB Dhruva, S. Sriram, Jay Lemmon, J. Merel, Greg Wayne, et al. Emergence of locomotion behaviours in rich environments. *ArXiv*, abs/1707.02286, 2017.

[52] Zhaoming Xie, G. Berseth, Patrick Clary, J. Hurst, and M. V. D. Panne. Feedback control for cassie with deep reinforcement learning. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246, 2018.

[53] L. Liu and J. Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37:1–14, 2018. 9, 37, 86

[54] X. Peng, P. Abbeel, Sergey Levine, and M. V. D. Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37:143:1–143:14, 2018. 12, 37

[55] S. Lee, Moonseok Park, K. Lee, and J. Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics (TOG)*, 38:1–13, 2019. 9, 37, 86

[56] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *Robotics: Science and Systems (RSS)*, 2020. 7, 9, 10, 38, 39, 40, 41, 42, 86

[57] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, volume abs/2004.00784, 07 2020. doi: 10.15607/rss.2020.xvi.064. 8, 9, 12, 13, 14, 16, 21, 32, 37, 38, 43, 47, 52, 54, 56, 84, 87, 92, 136, 137

[58] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd. 8, 11, 12, 41, 47, 54, 72, 73, 79

[59] B. Katz, J. Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, 2019. 9, 38, 52, 86

[60] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017. doi: 10.1177/0278364917694244. URL https://doi.org/10.1177/0278364917694244. 9, 38, 52, 86

[61] Zhaoming Xie, Patrick Clary, J. Dao, Pedro Morais, Jonanthan Hurst, and M. V. D. Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning (CoRL)*, 2019. 9, 37, 38, 52, 87

[62] J. Tan, Zhaoming Xie, Byron Boots, and C. Liu. Simulation-based design of dynamic controllers for humanoid balancing. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2729–2736, 2016. 9, 37, 52

[63] Yevgen Chebotar, A. Handa, Viktor Makoviychuk, M. Macklin, J. Issac, Nathan D. Ratliff, et al. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, 2019.

[64] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 1290–1296. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9562091. URL https://doi.org/10.1109/ICRA48506.2021.9562091. 9

[65] Fereshteh Sadeghi and Sergey Levine. CAD^{2}RL: Real single-image flight without a single real image. *Robotics: Science and Systems (RSS)*, 2017. 9, 37, 38, 87

[66] Zhanpeng He, Ryan C. Julian, Eric Heiden, H. Zhang, S. Schaal, Joseph J. Lim, et al. Zero-shot skill composition and simulation-to-real transfer by learning task representations. *ArXiv*, abs/1810.02422, 2018. 9, 38, 52

[67] Wenhao Yu, J. Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5:2950–2957, 2020. 9, 16, 37, 38, 87

[68] Ryan C. Julian, Benjamin Swanson, G. Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Efficient adaptation for end-to-end vision-based robotic manipulation. *ArXiv*, abs/2004.10190, 2020. 9

[69] Yuxiang Yang, K. Caluwaerts, Atil Iscen, T. Zhang, J. Tan, et al. Data efficient reinforcement learning for legged robots. *Conference on Robot Learning (CoRL)*, abs/1907.03613, 2019. 10, 37, 38, 39, 52

[70] Sungjoon Choi and J. Kim. Trajectory-based probabilistic policy gradient for learning locomotion behaviors. *2019 International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2019. 10, 37, 39, 52

[71] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan W. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. In Dylan A.

Shell, Marc Toussaint, and M. Ani Hsieh, editors, *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi: 10.15607/RSS.2021.XVII.061. URL https://doi.org/10.15607/RSS.2021.XVII.061. 11

[72] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018. 12, 24, 32, 39, 40, 41, 47, 69, 70, 73, 89, 91

[73] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 12, 91

[74] H. Zhang, S. Starke, T. Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37:1–11, 2018. 12

[75] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 12, 120, 121

[76] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org. 12

[77] J. Lee, Jemin Hwangbo, and M. Hutter. Robust recovery controller for a quadrupedal robot using deep reinforcement learning. *ArXiv*, abs/1901.07517, 2019. 14, 43, 84

[78] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 16

[79] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine learning*, 84(1-2):171–203, 2011. 21

[80] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[81] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[82] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. Pmlr, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/kalashnikov18a.html. 21, 68, 69

[83] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *International Conference on Intelligent Robots and Systems*, 2017. 21, 32

[84] C. Schenck and D. Fox. Visual closed-loop control for pouring liquids. In *International Conference on Robotics and Automation*, 2017. 21

[85] P. Kormushev, S. Calinon, and D. Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *International Conference on Intelligent Robots and Systems*, 2010. 21

[86] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 21, 32

[87] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. 21, 92

[88] Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. Preferences implicit in the state of the world. In *International Conference on Learning Representations*, 2019.

[89] Alexander Matt Turner, Neale Ratzlaff, and Prasad Tadepalli. Avoiding side effects in complex environments. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f50a6c02a3fc5a3a5d4d9391f05f3efc-Abstract.html. 21, 92

[90] Stefan Schaal. Learning from demonstration. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1997. URL https://proceedings.neurips.cc/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf. 21, 68, 87, 88, 97

[91] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[92] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

[93] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483, 2009. 21, 87

[94] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *International Conference on Advanced Robotics*, 2009. 21

[95] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *International Conference on Intelligent Robots and Systems*, 2011.

[96] B. Akgun, M. Cakmak, J. Yoo, and A. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *International Conference on Human-Robot Interaction*, 2012. 21

[97] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *International Conference on Knowledge Capture*, 2009. 22, 23

[98] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, David Roberts, Matthew E Taylor, et al. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*, 2017. 22, 23

[99] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007. 22, 26

[100] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010. 22, 26

[101] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. 23, 29

[102] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, et al. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020. 23, 29

[103] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, et al. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2020. 23, 29

[104] Patrick M Pilarski, Michael R Dawson, Thomas Degris, Farbod Fahimi, Jason P Carey, and Richard S Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *International Conference on Rehabilitation Robotics*, 2011. 23

[105] Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257*, 2019. 23

[106] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *International Conference on Robotics and Automation*, 2016. 23

[107] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, 2018.

[108] Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn. Few-shot goal inference for visuomotor learning and planning. In *Conference on Robot Learning*, 2018. 23

[109] Marvin Zhang, Sharad Vikram, Laura Smith, P. Abbeel, Matthew J. Johnson, et al. Solar: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019. 23, 50

[110] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In *Robotics: Science and Systems*, 2019. 32

[111] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *Robotics: Science and Systems*, 2020. 23, 117

[112] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Conference on Artificial Intelligence*, 2018. 23

[113] Riad Akrour, Marc Schoenauer, and Michele Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2011. 23

[114] Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012. 27

[115] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Advances in Neural Information Processing Systems*, 2012. 25

[116] Hiroaki Sugiyama, Toyomi Meguro, and Yasuhiro Minami. Preference-learning based inverse reinforcement learning for dialog control. In *Conference of the International Speech Communication Association*, 2012.

[117] Christian Wirth and Johannes Fürnkranz. Preference-based reinforcement learning: A preliminary survey. In *ECML/PKDD Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards*, 2013.

[118] Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. Model-free preference-based reinforcement learning. In *Conference on Artificial Intelligence*, 2016.

[119] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

[120] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *Conference on Robot Learning*, 2018.

[121] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[122] Erdem Biyik, Nicolas Huynh, Mykel J Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. In *Robotics: Science and Systems*, 2020. 23

[123] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015. 23, 27

[124] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, et al. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016. 23, 27

[125] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, 2018. 24, 26, 27, 29, 33, 121

[126] Zehong Cao, KaiChiu Wong, and Chin-Teng Lin. Human preference scaling with demonstrations for deep reinforcement learning. *arXiv preprint arXiv:2007.12904*, 2020. 24

[127] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *The Journal of Machine Learning Research*, 17(1): 3190–3239, 2016. 24

[128] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2018.

[129] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[130] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. 24

[131] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. 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=HJgLZR4KvH. 24

[132] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, 2016. 24

[133] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

[134] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. 24

[135] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016. 24

[136] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, et al. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[137] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, 2017. 24

[138] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, 2019. 24, 26

[139] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[140] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18459–18473, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/99bf3d153d4bf67d640051a1af322505-Abstract.html. 24, 26, 27

[141] Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, USA, 2010. URL https://doi.org/10.1184/r1/6720692.v1. 24

[142] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 25

[143] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, 2021. 26

[144] Jan Beirlant, Edward J Dudewicz, László Györfi, and Edward C Van der Meulen. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997. 26, 119

[145] Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23(3-4):301–321, 2003. 26, 119

[146] Leonard J Savage. *The foundations of statistics*. Courier Corporation, 1972. 27

[147] Paolo Viappiani. Monte carlo methods for preference learning. In *International Conference on Learning and Intelligent Optimization*, 2012. 27

[148] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. In *Robotics: Science and Systems*, 2014. 27

[149] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 31

[150] A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, 2017. 32

[151] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *Conference on Robot Learning (CoRL)*, abs/1806.10293, 2018. 36, 50

[152] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37:421–436, 2018. 50

[153] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 885–897. PMLR, 2019. URL http://proceedings.mlr.press/v100/dasari20a.html.

[154] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, et al. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv*, abs/2104.08212, 2021. 50

[155] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In Kris Hauser, Dylan A. Shell, and Shoudong Huang, editors, *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*, 2022. doi: 10.15607/RSS.2022.XVIII.063. URL https://doi.org/10.15607/RSS.2022.XVIII.063. 36, 50

[156] Mark Cutler, Thomas J. Walsh, and Jonathan P. How. Reinforcement learning with multi-fidelity simulators. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3888–3895, 2014. 37, 87

[157] Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations (ICLR)*, abs/1610.01283, 2017. 37, 87, 88

[158] Jane X. Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z. Leibo, Dhruva Tirumala, et al. Learning to reinforcement learn. *ArXiv*, abs/1611.05763, 2017. 37, 87

[159] Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

[160] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *International Conference on Machine Learning (ICML)*, abs/1903.08254, 2019.

[161] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, et al. Rapidly adaptable legged robots via evolutionary meta-learning. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776, 2020. 37, 87

[162] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and P. Abbeel. Daydreamer: World models for physical robot learning. *Conference on Robot Learning (CoRL)*, abs/2206.14176, 2022. 37, 38, 39, 40, 42, 53, 54

[163] Gavin D. Kenneally, Avik De, and Daniel E. Koditschek. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1: 900–907, 2016. 38

[164] Yuxiang Yang, Tingnan Zhang, Erwin Coumans, Jie Tan, and Byron Boots. Fast and efficient locomotion via learned gait transitions. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 773–783. PMLR, 2021. URL `https://proceedings.mlr.press/v164/yang22d.html`. 38, 52

[165] Kevin Sebastian Luck, Joseph Campbell, Michael Andrew Jansen, Daniel M. Aukes, and Heni Ben Amor. From the lab to the desert: Fast prototyping and learning of robot locomotion. *Robotics: Science and Systems (RSS)*, abs/1706.01977, 2017. 39, 52

[166] Tsung-Yen Yang, Tingnan Zhang, Linda Luu, Sehoon Ha, Jie Tan, and Wenhao Yu. Safe reinforcement learning for legged locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October*

23-27, 2022, pages 2454–2461. IEEE, 2022. doi: 10.1109/IROS47612.2022.9982038. URL https://doi.org/10.1109/IROS47612.2022.9982038. 39, 91

[167] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *International Conference on Learning Representations (ICLR)*, 2020. 39, 54

[168] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, et al. Massively parallel methods for deep reinforcement learning. *ArXiv*, abs/1507.04296, 2015. 40

[169] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. In *International Conference on Learning Representations (ICLR)*, 2017.

[170] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html. 40

[171] Shixiang Shane Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, 2017. 40

[172] Yunzhi Zhang, Ignasi Clavera, Bo-Yu Tsai, and P. Abbeel. Asynchronous methods for model-based reinforcement learning. *Conference on Robot Learning (CoRL)*, abs/1910.12453, 2019. 40

[173] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 40

[174] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xCVJMsPv3RT. 40, 41, 47, 54, 72, 73, 79, 80, 91

[175] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014. 40, 41, 54

[176] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, abs/1607.06450, 2016. doi: 10.48550/arxiv.1607.06450. URL https://arxiv.org/abs/1607.06450. 40, 41, 54, 67, 71, 91

[177] Yi-Han Xu, Cheng-Cheng Yang, Min Hua, and Wen Zhou. Deep deterministic policy gradient (ddpg)-based resource allocation scheme for noma vehicular communications. *IEEE Access*, 8:18797–18807, 2020. 41, 89

[178] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, et al. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 42, 54, 82, 91

[179] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, pages 5026–5033. Ieee, 2012. doi: 10.1109/iros.2012.6386109. 42

[180] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, et al. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099. 42, 43

[181] Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. *Conference on Robot Learning (CoRL)*, 2021. 43, 54, 56, 84, 86

[182] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey 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. 50

[183] J. Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, P. Pastor, et al. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40:698–721, 2021. 50

[184] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793, 2017. 50

[185] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron C. Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:248811264. 52, 54

[186] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael N. Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. *2010 IEEE International Conference on Robotics and Automation*, pages 2665–2670, 2010. 52, 84, 86

[187] Dario Bellicoso, Fabian Jenelten, Christian Gehring, and Marco Hutter. Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robotics and Automation Letters*, 3:2261–2268, 2018. 84, 86, 89

[188] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. Ieee, 2014. 86, 89

[189] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, et al. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016. 52, 86

[190] Yanran Ding, Abhishek Pandala, Chuanzheng Li, Young-Ha Shin, and Hae won Park. Representation-free model predictive control for dynamic motions in quadrupeds. *IEEE Transactions on Robotics*, 37:1154–1171, 2020. URL https://api.semanticscholar.org/CorpusID:229331611. 52

[191] Gabriel B. Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *Int. J. Robotics Res.*, 43(4):572–587, 2024. doi: 10.1177/02783649231224053. URL https://doi.org/10.1177/02783649231224053. 52, 87

[192] N. Rudin, Hendrik Kolvenbach, Vassilios Tsounis, and Marco Hutter. Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning. *IEEE Transactions on Robotics*, 38:317–328, 2021. 52, 86

[193] G. Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, et al. Learning to jump from pixels. In *Conference on Robot Learning*, 2021. 52, 86

[194] Chenxiao Yu and Andre Rosendo. Multi-modal legged locomotion framework with automated residual reinforcement learning. *IEEE Robotics and Automation Letters*, 7:10312–10319, 2022. 52, 86

[195] Eric Vollenweider, Marko Bjelonic, Victor Klemm, Nikita Rudin, Joonho Lee, and Marco Hutter. Advanced skills through multiple adversarial motion priors in reinforcement learning. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5120–5126. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160751. URL https://doi.org/10.1109/ICRA48891.2023.10160751. 86

[196] Yuni Fuchioka, Zhaoming Xie, and Michiel van de Panne. Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5092–5098. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160562. URL https://doi.org/10.1109/ICRA48891.2023.10160562. 52, 86, 87

[197] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron C. Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:259298604. 54

[198] Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/deepmind/mujoco%5Fmenagerie. 54

[199] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12498–12509, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html. 54

[200] Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. Regularization matters in policy optimization - an empirical study on continuous control. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=yr1mzrH3IC. 54

[201] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6131–6141. PMLR, 2021. URL http://proceedings.mlr.press/v139/lee21g.html. 54

[202] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=14-kr46GvP-. 54, 72

[203] Pranav Shyam, Wojciech Jaśkowski, and Faustino J. Gomez. Model-based active exploration. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:53102049. 57

[204] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim M. Songhori, Shen Wang, et al. A graph placement methodology for fast chip design. *Nature*, 594 7862:207–212, 2021. 66

[205] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON. 66

[206] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643. 66, 68, 69

[207] Andrew Wagenmaker and Aldo Pacchiano. Leveraging offline data in online reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 35300–35338. PMLR, 2023. URL https://proceedings.mlr.press/v202/wagenmaker23a.html. 66

[208] Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yyBis80iUuU. 66, 69

[209] Serkan Cabi, Sergio Gomez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott E. Reed, Rae Jeong, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *Robotics: Science and Systems XVI*, 2019. 66

[210] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv*, June 2020. 66, 68, 75, 76, 133

[211] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, et al. Aw-opt: Learning robotic skills with imitation andreinforcement at scale. In *5th Annual Conference on Robot Learning*, 2021. 66

[212] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005. URL http://jmlr.org/papers/v6/ernst05a.html. 68

[213] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. Pmlr, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html. 68, 69

[214] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, et al. Deep q-learning from demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11757. URL https://ojs.aaai.org/index.php/AAAI/article/view/11757. 68, 71

[215] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=AlJXhEI6J5W. 68, 71, 76, 131

[216] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8. 68, 76, 133

[217] Sergey Levine and Vladlen Koltun. Guided policy search. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1–9, Atlanta, Georgia, USA, 17–19 Jun 2013. Pmlr. URL https://proceedings.mlr.press/v28/levine13.html. 68

[218] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

[219] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299, 2017. doi: 10.1109/icra.2018.8463162. URL https://doi.org/10.1109/ICRA.2018.8463162. 68, 69, 71, 88, 97

[220] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, volume abs/1709.10087, 2018. 68

[221] Tim G. J. Rudner, Cong Lu, Michael A. Osborne, Yarin Gal, and Yee Whye Teh. On pathologies in kl-regularized reinforcement learning from expert demonstrations. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28376–28389, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/eecca5b6365d9607ee5a9d336962c534-Abstract.html. 68, 71, 88, 97, 133

[222] Haruhiko Asada and Hideo Hanafusa. Playback control of force teachable robots. *Transactions of the Society of Instrument and Control Engineers*, 15(3):410–411, 1979. doi: 10.9746/sicetr1965.15.410. 68, 88

[223] Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017. URL http://arxiv.org/abs/1707.08817. 68, 72, 76, 79, 88, 91

[224] Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=JdTnc9gjVfJ. 69

[225] Haichao Zhang, Wei Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-Y34L45JR6z. 69

[226] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518. 69, 88

[227] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, et al. Soft actor-critic algorithms and applications, 2018. URL https://arxiv.org/abs/1812.05905. 69, 73

[228] Stéphane Ross and J. Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, page 1905–1912, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851. 70

[229] Sebastian Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of 4th Connectionist Models Summer School*. Erlbaum Associates, June 1993. 70

[230] Shideh Rezaeifar, Robert Dadashi, Nino Vieillard, Léonard Hussenot, Olivier Bachem, Olivier Pietquin, et al. Offline reinforcement learning as anti-exploration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):8106–8114, Jun. 2022. doi: 10.1609/aaai.v36i7.20783. URL https://ojs.aaai.org/index.php/AAAI/article/view/20783. 71

[231] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html. 72

[232] Edoardo Cetin, Philip J Ball, Stephen Roberts, and Oya Celiktutan. Stabilizing off-policy deep reinforcement learning from pixels. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2784–2810. Pmlr, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/cetin22a.html. 72

[233] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf. 72

[234] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=%5FSJ-%5Fyyes8. 72, 77

[235] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694. 72

[236] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, et al. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1etN1rtPB.

[237] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, et al. What matters for on-policy deep actor-critic methods? a

large-scale study. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nIAxjsniDzg.

[238] Hiroki Furuta, Tadashi Kozuno, Tatsuya Matsushima, Yutaka Matsuo, and Shixiang Gu. Co-adaptation of algorithmic and implementational innovations in inference-based deep reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=vLyI%5F%5FSoeAe. 72, 73

[239] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. doi: 10.1609/aaai.v30i1.10295. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295. 73

[240] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html. 73

[241] Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12849–12863. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/6abcc8f24321d1eb8c95855eab78ee95-Paper.pdf. 73

[242] Philip J. Ball and Stephen J. Roberts. Offcon^3: What is state of the art anyway?, 2021. URL https://arxiv.org/abs/2101.11331. 73

[243] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020. 76

[244] Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. In *Workshop on Learning from Diverse, Offline Data*, 2022. URL https://openreview.net/forum?id=bPOBIKaqLba. 76

[245] Marc H. Raibert. Hopping in legged systems – modeling and simulation for the two-dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics*, Smc-14:451–463, 1984. 84

[246] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *The International Journal of Robotics Research*, 3:60–74, 1984.

[247] Koushil Sreenath, Hae won Park, Ioannis Poulakakis, and Jessy W. Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on mabel. *The International Journal of Robotics Research*, 32: 324–345, 2013.

[248] Christian M. Hubicki, Andy Abate, Patrick Clary, Siavash Rezazadeh, Mikhail S. Jones, Andrew Peekema, et al. Walking and running with passive compliance: Lessons from engineering: A live demonstration of the atrias biped. *IEEE Robotics & Automation Magazine*, 25:23–39, 2018.

[249] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *ArXiv*, abs/1909.06586, 2019. 86, 89

[250] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors. *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2021. 84

[251] Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018. 86, 89

[252] Will Bosworth, Jonas Whitney, Sangbae Kim, and Neville Hogan. Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3582–3589. Ieee, 2016.

[253] Ludovic Righetti and Stefan Schaal. Quadratic programming for inverse dynamics with optimal distribution of contact forces. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 538–543. Ieee, 2012. 86

[254] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Sci. Robotics*, 5(47): 5986, 2020. doi: 10.1126/SCIROBOTICS.ABC5986. URL https://doi.org/10.1126/scirobotics.abc5986. 86

[255] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. *Conference on Robot Learning (CoRL)*, 2022. 86, 91

[256] Hae won Park, Patrick M. Wensing, and Sangbae Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In *Robotics: Science and Systems*, 2015. 86

[257] Hae-Won Park, Patrick M. Wensing, and Sangbae Kim. Jumping over obstacles with MIT cheetah 2. *Robotics Auton. Syst.*, 136:103703, 2021. doi: 10.1016/J.ROBOT.2020.103703. URL https://doi.org/10.1016/j.robot.2020.103703. 86

[258] Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. ALLSTEPS: curriculum-driven learning of stepping stone skills. *Comput. Graph. Forum*, 39(8):213–224, 2020. doi: 10.1111/CGF.14115. URL https://doi.org/10.1111/cgf.14115. 86

[259] Guillaume Bellegarda, Chuong Nguyen, and Quan Nguyen. Robust quadruped jumping via deep reinforcement learning. *Robotics Auton. Syst.*, 182:104799, 2024. doi: 10.1016/J.ROBOT.2024.104799. URL https://doi.org/10.1016/j.robot.2024.104799. 86

[260] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, et al. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5:3650–3657, 2019. 86

[261] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311. 87, 92

[262] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37 (6), November 2018.

[263] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38(6):1–11, 2019.

[264] Levi Fussell, Kevin Bergamin, and Daniel Holden. Supertrack: Motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021.

[265] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4):33–1, 2020. 87, 92

[266] Xue Bin Peng, Ze Ma, P. Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, 40:144:1–144:20, 2021. 87

[267] Alejandro Escontrela, Xue Bin Peng, Wenhao Yu, Tingnan Zhang, Atil Iscen, Ken Goldberg, et al. Adversarial motion priors make good substitutes for complex reward functions. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 25–32, 2022. 87

[268] Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization. *Frontiers in Robotics and AI*, 9, 2021. 87

[269] Atil Iscen, George Yu, Alejandro Escontrela, Deepali Jain, Jie Tan, and Ken Caluwaerts. Learning agile locomotion skills with a mentor. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 2019–2025. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561567. URL https://doi.org/10.1109/ICRA48506.2021.9561567. 87, 91

[270] Yujin Tang, Jie Tan, and Tatsuya Harada. Learning agile locomotion via adversarial training. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6098–6105, 2020. 87

[271] Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, et al. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2017. 88, 97

[272] Annie Xie and Chelsea Finn. Lifelong robotic reinforcement learning by retaining experiences. In Sarath Chandar, Razvan Pascanu, and Doina Precup, editors, *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pages 838–855. PMLR, 2022. URL https://proceedings.mlr.press/v199/xie22a.html. 88

[273] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018. 92

[274] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language

model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 2023. URL https://proceedings.mlr.press/v202/driess23a.html. 103

[275] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, et al. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. 103

[276] Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pages 496–512. PMLR, 2024. URL https://proceedings.mlr.press/v270/doshi25a.html. 103

[277] Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011. ISBN 9780374275631 0374275637. URL https://www.amazon.de/Thinking-Fast-Slow-Daniel-Kahneman/dp/0374275637/ref=wl%5Fit%5Fdp%5Fo%5FpdT1%5FnS%5FnC?ie=UTF8&colid=151193SNGKJT9&coliid=I3OCESLZCVDFL7. 103

[278] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022. 103

[279] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, et al. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. Ieee, 2023. 103, 105

[280] Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado, editors, *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.096. URL https://doi.org/10.15607/RSS.2024.XX.096. 103

[281] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad (Mahi) Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=hoVwecMqV5. 104

[282] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):11205–11212, 2022. 104

[283] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288. 104, 110

[284] OpenAI et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[285] Gemini Team. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805. 104

[286] Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, Chelsea Finn, and Karol Hausman. Open-world object manipulation using pre-trained vision-language models. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pages 3397–3417. PMLR, 2023. URL https://proceedings.mlr.press/v229/stone23a.html. 104, 109

[287] Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, et al. Robotic skill acquistion via instruction augmentation with vision-language models. In *Proceedings of Robotics: Science and Systems*, 2023. 104

[288] Vivek Myers, Chunyuan Zheng, Oier Mees, Kuan Fang, and Sergey Levine. Policy adaptation via language optimization: Decomposing tasks for few-shot imitation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pages 1402–1426. PMLR, 2024. URL https://proceedings.mlr.press/v270/myers25a.html. 104

[289] Jesse Zhang, Karl Pertsch, Jiahui Zhang, and Joseph J Lim. Sprint: Scalable policy pre-training via language instruction relabeling. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9168–9175. Ieee, 2024. 104

[290] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019. 105

[291] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Rich Walke, Chelsea Finn, Aviral Kumar, et al. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *The Twelfth International Conference on Learning Rep-*

resentations, volume abs/2310.10639, 2023. URL https://api.semanticscholar.org/CorpusID:264172455. 105

[292] Kuan Fang, Patrick Yin, Ashvin Nair, and Sergey Levine. Planning to practice: Efficient online fine-tuning by composing goals in latent space. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4076–4083, 2022. URL https://api.semanticscholar.org/CorpusID:248834175. 105

[293] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. In *Robotics: Science and Systems (RSS)*, 2024. 105

[294] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, et al. Any-point trajectory modeling for policy learning. *Robotics: Science and Systems (RSS)*, 2024. 105

[295] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, et al. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023. 105

[296] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. In *Forty-first International Conference on Machine Learning*, 2024. 105

[297] Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2019. 105

[298] Youngwoon Lee, Jingyun Yang, and Joseph J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. 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=ryxB2lBtvH.

[299] Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663. Pmlr, 2018.

[300] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2019.

[301] Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, pages 188–204. Pmlr, 2020. 105

[302] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2021.

[303] Mohit Sharma, Jacky Liang, Jialiang Zhao, Alex LaGrassa, and Oliver Kroemer. Learning to compose hierarchical object-centric controllers for robotic manipulation. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 822–844. PMLR, 2020. URL https://proceedings.mlr.press/v155/sharma21a.html.

[304] Murtaza Dalal, Deepak Pathak, and Ruslan Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21847–21859, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/b6846b0186a035fcc76b1b1d26fd42fa-Abstract.html. 105

[305] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[306] Jesse Zhang, Minho Heo, Zuxin Liu, Erdem Biyik, Joseph J Lim, Yao Liu, and Rasool Fakoor. EXTRACT: Efficient policy learning by extracting transferrable robot skills from offline data. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=uEbJXWobif. 105

[307] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1726–1734. AAAI Press, 2017. doi: 10.1609/AAAI.V31I1.10916. URL https://doi.org/10.1609/aaai.v31i1.10916.

[308] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3307–3317, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

[309] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32, 2019. 105

[310] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *International Conference on Computer Vision (ICCV)*, 2019. 105

[311] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 3272–3282. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00328. URL https://doi.org/10.1109/CVPR52688.2022.00328. 105

[312] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 1–13. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01324. URL https://doi.org/10.1109/CVPR52729.2023.01324. 105

[313] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.012. URL https://doi.org/10.15607/RSS.2023.XIX.012. 105

[314] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *Advances in Neural Information Processing Systems*, 36, 2024. 106

[315] Jeremy Morton and Mykel J. Kochenderfer. Simultaneous policy learning and latent state inference for imitating driver behavior. In *20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017*, pages 1–6. IEEE, 2017. doi: 10.1109/ITSC.2017.8317738. URL https://doi.org/10.1109/ITSC.2017.8317738. 106

[316] Sravan Jayanthi, Letian Chen, Nadya Balabanska, Van Duong, Erik Scarlatescu, Ezra Ameperosa, et al. Droid: Learning from offline heterogeneous demonstrations via reward-policy distillation. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1547–1571. Pmlr, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/jayanthi23a.html. 106

[317] Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav S. Sukhatme, and Joseph J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1235–1245, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/632cee946db83e7a52ce5e8d6f0fed35-Abstract.html. 106

[318] Montserrat Gonzalez Arenas, Ted Xiao, Sumeet Singh, Vidhi Jain, Allen Z Ren, Quan Vuong, et al. How to prompt your robot: A promptbook for manipulation skills with code as policies. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. 107

[319] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530. 109

[320] O X E Collaboration et al. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024. 110

[321] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: action hierarchies using language. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado, editors, *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.049. URL https://doi.org/10.15607/RSS.2024.XX.049. 111

[322] Russell Mendonca, Emmanuel Panov, Bernadette Bucher, Jiuguang Wang, and Deepak Pathak. Continuously improving mobile manipulation with autonomous real-world RL. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pages 5204–5219. PMLR, 2024. URL https://proceedings.mlr.press/v270/mendonca25a.html. 116, 117

[323] Kyle Stachowicz, Dhruv Shah, Arjun Bhorkar, Ilya Kostrikov, and Sergey Levine. Fastrlap: A system for learning high-speed driving via deep RL and autonomous practicing. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pages 3100–3111. PMLR, 2023. URL https://proceedings.mlr.press/v229/stachowicz23a.html. 116

[324] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A software

suite for sample-efficient robotic reinforcement learning. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pages 16961–16969. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610040. URL https://doi.org/10.1109/ICRA57147.2024.10610040. 116

[325] Abhishek Gupta, Justin Yu, Tony Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, et al. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *ArXiv*, abs/2104.11203, 2021. 117

[326] Zheyuan Hu, Aaron Rovinsky, Jianlan Luo, Vikash Kumar, Abhishek Gupta, and Sergey Levine. REBOOT: reuse data for bootstrapping efficient real-world dexterous manipulation. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pages 1930–1949. PMLR, 2023. URL https://proceedings.mlr.press/v229/hu23a.html. 117