Learning and Adapting Agile Locomotion Skills by Transferring Experience

Laura Smith, J. Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, Jie Tan, Sergey Levine
Berkeley Artificial Intelligence Research, Berkeley, CA, 94720
Email: smithlaura@berkeley.edu

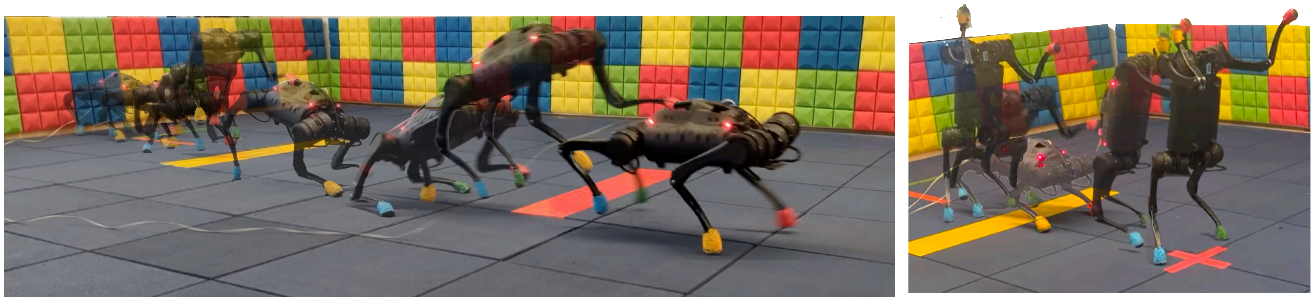


Fig. 1: **Agile skills** learned with our proposed method enable the A1 robot to jump repeatedly (**left**) and walk to a goal location on its hind legs (**right**).

***Abstract*—Legged robots have enormous potential in their range of capabilities, from navigating unstructured terrains to high-speed running. However, designing robust controllers for highly agile dynamic motions remains a substantial challenge for roboticists. Reinforcement learning (RL) offers a promising data-driven approach for automatically training such controllers. However, exploration in these high-dimensional, underactuated systems remains a significant hurdle for enabling legged robots to learn performant, naturalistic, and versatile agility skills. We propose a framework for training complex robotic skills by transferring experience from existing controllers to jumpstart learning new tasks. To leverage controllers we can acquire in practice, we design this framework to be flexible in terms of their source—that is, the controllers may have been optimized for a different objective under different dynamics, or may require different knowledge of the surroundings—and thus may be highly suboptimal for the target task. We show that our method enables learning complex agile jumping behaviors, navigating to goal locations while walking on hind legs, and adapting to new environments. We also demonstrate that the agile behaviors learned in this way are graceful and safe enough to deploy in the real world. (Videos[1])**

I. Introduction

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 robots, but they often require extensive knowledge of the robot and environment dynamics, as well as task-specific knowledge about the desired agile movement [1–8].

[1]https://sites.google.com/berkeley.edu/twirl

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 [9–12]. 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 1 shows a task where the robot needs to walk to a

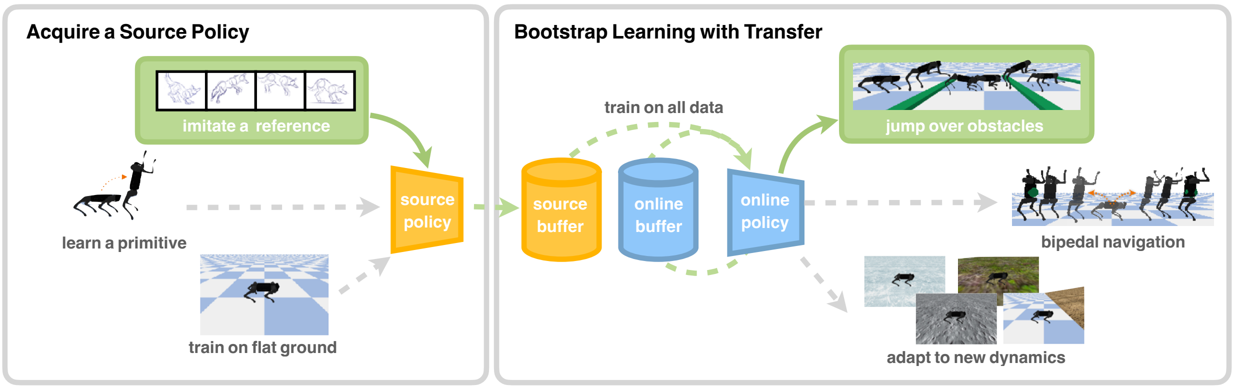


Fig. 2: Overview of three different applications of our system. We identify that several fundamental challenges in learning agile locomotion skills can be ameliorated by casting them as transfer learning problems, then applying a simple, generic method that involves a simple modification to off-the-shelf 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.

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.~~Adapting to new environments looks very similar—if we have an agile skill, faced with a new environment we should be able to adapt it rather than try to learn from scratch.already have a skill that performs an agile behavior, we might similarly prefer to initialize from the existing skill and then fine-tune to the new setting.~~ 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 of this paper 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 IV, 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.

II. Related Work

Developing control strategies for legged robots has been a long-standing research topic in robotics. Model-based methods that leverage trajectory optimization [3, 5] and model-predictive control (MPC) [13–18] tackle this problem by using manually constructed dynamics models of the robot. These methods have shown compelling results on robust control of multiple gaits [7] and adaptation to complex terrains [19–21]. 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 [22–24] and even in the real world [25–34].

In this work, 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 [35, 36], MiniCheetah [7] and A1 [37] 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 [38, 39]. As for fully learning-based approaches, Rudin et al. [40] learn a jumping controller focused on reorientation and landing in micro-gravity environments. In terms of comparable results to ours, Margolis et al. [39] demonstrate that using high-level velocity and foot force commands can enable a~~that they are able to get the~~ MiniCheetah to continuously hop over randomly placed gaps.~~ but to do so they utilize high-level velocity and foot force commands that are then passed to a low-level MPC controller to realize them~~. When transferred to the real world, the robot is~~In the real world, they are~~ 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. [41], 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 [42] 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. [43] use RL to imitate a reference motion of bipedal stepping generated with trajectory optimization on a Solo 8 robot [44]. 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 or broader tasks.

As mentioned in Section I, the primary challenges in learning high dimensional underactuated robotic systems are task specification and exploration. A popular way to address these two challenges is imitation learning [45–48], where the agent is trained to copy an expert. In locomotion, this principle often manifests as tracking reference motions [43, 49–54]. 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 [55, 56]. Bogdanovic et al. [57] 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. [58] 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. [59] 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 [60, 61]. 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 [11, 62–70] 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 [71–74]. 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 [75–80], 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 [81] and Nair et al. [78], 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 [82] employ a similar methodology to ours in the continual learning setting while assuming functional access to the reward function to relabel past experiences with the new task and filtering to only continue learning on transitions that are similar to the online data and demonstrate results on a sequence of manipulation tasks with end-effector control at 5Hz. In comparison, we study the ability to apply this principle to very difficult locomotion skills with low-level control at 20Hz.

III. Preliminaries

We frame learning a locomotion skill as a Markov decision process (MDP) [83] $\mathcal{M}$, defined as a tuple $(\mathcal{S}, \mathcal{A}, \gamma, p, r, \mu_0)$, 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: $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$.

a) 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.

b) 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 [84, 85] 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.

IV. 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 I 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.

A. 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 [5, 7, 18, 19]. To handle all these cases, a key desideratum of our system is it should be agnostic to *how* the source policy is obtained.

Algorithm 1 TWiRL pseudocode

```
Require: Source policy $\pi_{\text{src}}$, MDP $\mathcal{M}_{\text{src}}$ , MDP $\mathcal{M}_{\text{target}}$ of the desired task
 1: Initialize: Source policy replay buffer $\mathcal{D}_{\text{src}}$, online replay buffer $\mathcal{D}_{\text{target}}$,
    sampling ratio $\phi$, policy parameters $\theta$

 2: // OBTAIN DATA FROM $\pi_{\text{SRC}}$
 3: if $\mathcal{D}_{\text{src}}$ is empty then
 4:     repeat
 5:         Collect data with $\pi_{\text{src}}(\mathbf{a} \mid \mathbf{s})$.
 6:         Add experience to source buffer $\mathcal{D}_{\text{src}} \leftarrow (\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 7:     until desired

 8: // TRAIN ON ALL DATA USING HIGH UTD OFF-POLICY RL
 9: repeat
10:     Collect data with $\pi_\theta(\mathbf{a} \mid \mathbf{s})$.
11:     Add experience to online buffer $\mathcal{D} \leftarrow (\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
12:     Construct $\beta_{\text{src}}$ sampling $\phi$·batch size from $\mathcal{D}_{\text{src}}$
13:     Construct $\beta_{\text{target}}$ sampling $(1-\phi)$·batch size from $\mathcal{D}$
14:     Update $\theta$ using $\beta_{\text{src}} \cup \beta_{\text{target}}$
15: until converged
```

B. 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 1) 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 1, line 14) and data collection (line 10). This was also observed by Vecerík et al. [81], 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 [86] sufficient. Intuitively, taking more gradient updates may be important for effectively processing and incorporating the source policy data.

V. 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.

A. Setup Details

We use the A1 robot from Unitree as our platform and build our simulation using PyBullet [87]. 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 [10, 25, 88, 89] 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 IV-B) is a state-of-the-art off-policy actor-critic algorithm DroQ [86], a variant of SAC [84] that incorporates dropout regularization and layer normalization [90]. We use the open-sourced JAX [91] implementation from Smith et al. [92]. We report the average return and standard deviation across 3 runs unless otherwise stated.

B. Applications

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 [34, 60]. 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,

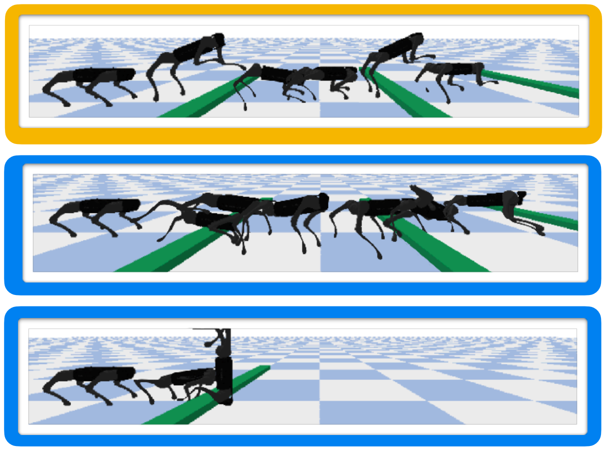


Fig. 3: Examples of our policy (outlined in yellow), which incorporates both online training and data from a motion imitation policy, compared to two policies (outlined in blue) trained from scratch with the same reward function. While naïvely optimizing for the task either exploits the simulator to learn an unnatural motion (middle) or fails completely (bottom), the policy trained by incorporating prior data exhibits a graceful jump.

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:

$$r(\mathbf{s},\mathbf{a})^{\text{jump}} = r^{\text{rv}}(\mathbf{s},\mathbf{a}) + r^{\text{o}}(\mathbf{s},\mathbf{a}) + r^{\text{jv}}(\mathbf{s},\mathbf{a}) + r^{\text{p}}(\mathbf{s},\mathbf{a}), \quad (1)$$

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 V-A, 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 3, training directly with the task reward in this case has two failure modes. First, we observe reward exploitation [93–95], 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

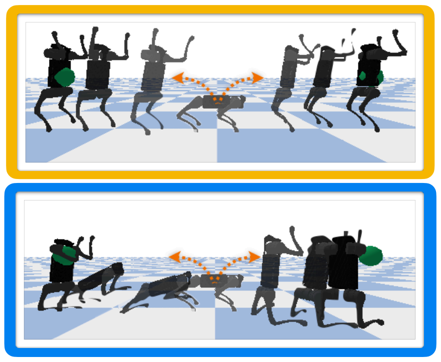


Fig. 4: Examples of our policy (outlined in yellow), trained with data from a robot that can already stand on its hind legs, compared to a baseline policy (blue) trained from scratch. Without this added bias, the baseline policy learns to scoot toward the goal on its knees. Our policy gracefully kicks up to standing and navigates to the goal on 2 legs.

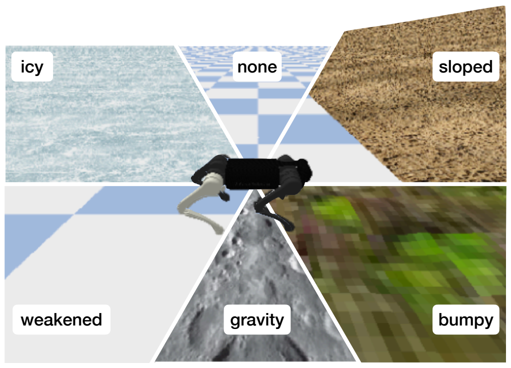


Fig. 5: Illustrating the diverse environments to which we adapt the source policy (trained in the environment labeled 'none' to indicate no modification). In clockwise order: the default, non-randomized environment; a sloped terrain; bumpy terrain; a low-gravity environment; a stochastic environment simulating motor weakening; a simulated ice rink.

shaped reward functions [49–54]. 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. [11] to learn a policy that imitates a leaping motion taken from the dataset provided by Zhang et al. [96]. 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 VI, we demonstrate that the jumping task can be solved perfectly by incorporating this data via TWiRL.

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 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:

$$r(\mathbf{s}, \mathbf{a}) = r^{\text{s}}(\mathbf{s}, \mathbf{a}) \cdot (1 + r^{\text{f}}(\mathbf{s}, \mathbf{a}) + r^{\text{d}}(\mathbf{s}, \mathbf{a})). \tag{2}$$

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 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 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 $\mathcal{M}_{\text{target}}$.

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 ~~it is~~ 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 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.

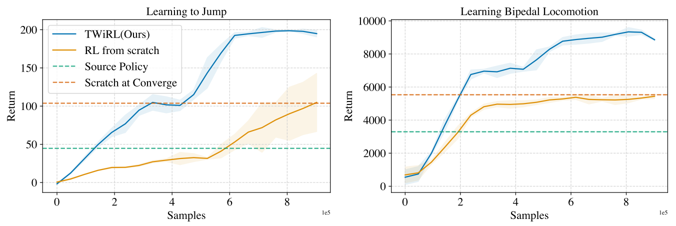


Fig. 6: Learning curves for the jumping (left) and bipedal navigation (right) tasks comparing learning from scratch (solid yellow curve) to TWiRL (solid blue curve) using data from a source policy (dotted green line), shown to 1M steps. Since the policies from scratch had not yet converged for the jumping 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.

VI. Results

We now present our experimental analysis of TWiRL. As a baseline, we evaluate directly applying RL to solve the tasks introduced in Section V 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 V-B (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 6). For the jumping task (the left plot), TWiRL (blue) consistently achieves **2**× 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 3, blue). As shown in the second plot of Figure 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 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 V, for jumping, we use a

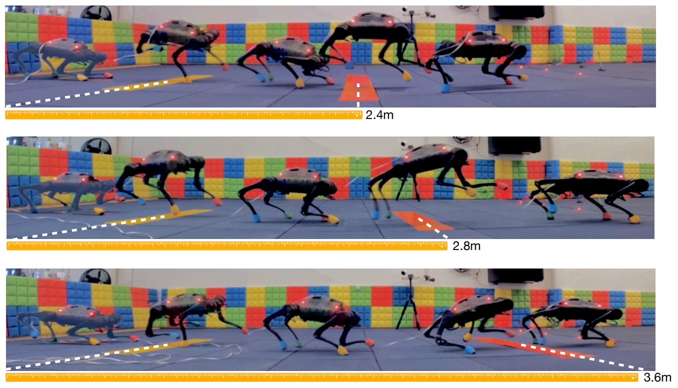
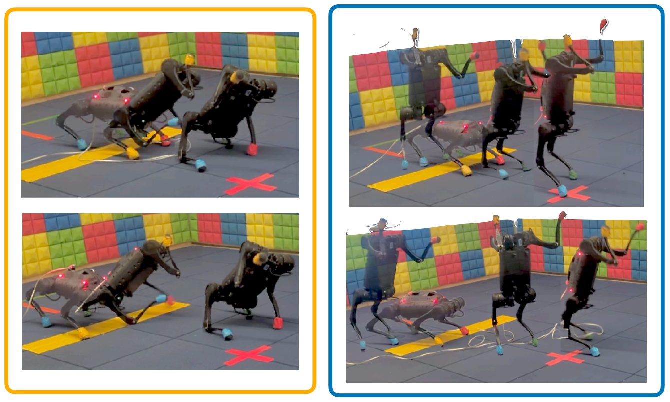


Fig. 7: Example successful rollouts from evaluating our jumping policy in the real world with different spacings of hurdles. From top to bottom the spacing is at 2.4m, 2.8m, and 3.6m, with success rates over 8 trials of **75%, 100%, and 75%**, respectively. We see that our policy exhibits the desired 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 in the supplementary material.



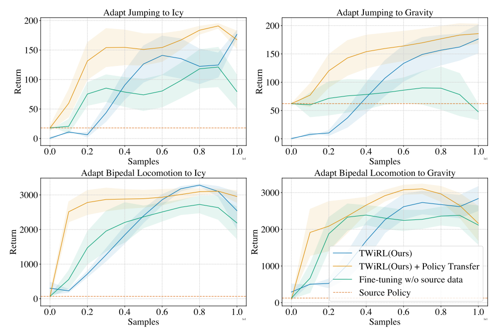
(a) Examples a policy learned from scratch (left, yellow) and learned by TWiRL (right, blue) rolled out in the real world. Videos of all evaluation trials can be found in the supplementary material.

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

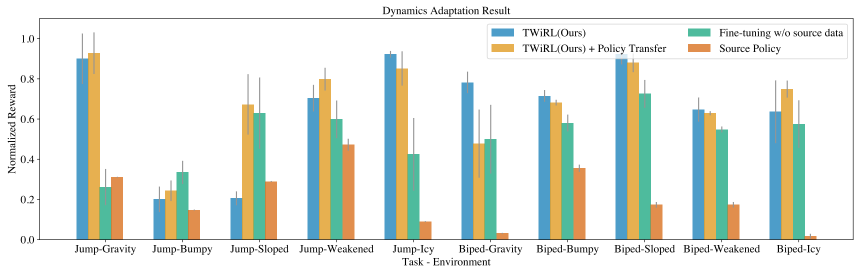
(b) We report the time the robot remained standing, the minimum distance the robot got to the goal, and the return as defined by our task.

Fig. 8: 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.

policy trained to 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



(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).

Fig. 9: Evaluating TWiRL in adapting the jumping and bipedal navigation skills to 5 different environments (see Section V) 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. Please refer to the supplementary material for all learning curves.

dashed green lines in Figure 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 3 and Figure 4). This is not guaranteed since, unlike other works [75–80], 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 the project website 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 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). 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 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 knees and one forepaw. ~~To our knowledge, neither of our results has been shown before from a learning-based method, on a hardware robot.~~ 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 V. 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 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 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.

VII. 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 pre-training. 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.

VIII. Acknowledgments

This work was supported in part by ARO W911NF-21-1-0097, the Office of Naval Research, and Google. Laura Smith is supported by NSF Graduate Research Fellowship. We thank Philipp Wu, Kevin Zakka, Dhruv Shah, Qiyang Li, Katie Kang, and Dibya Ghosh for their valuable feedback.

References

[1] 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. 1

[2] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *The International Journal of Robotics Research*, 3:60 – 74, 1984.

[3] 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. 2

[4] 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.

[5] 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. 2, 4

[6] Christian M. Hubicki, Andy Abate, Patrick Clary, Siavash Rezazadeh, Mikhail S. Jones, Andrew Peekema, Johnathan Van Why, Ryan Domres, Albert Wu, William C. Martin, Hartmut Geyer, and Jonathan W. Hurst. Walking and running with passive compliance: Lessons from engineering: A live demonstration of the atrias biped. *IEEE Robotics & Automation Magazine*, 25:23–39, 2018.

[7] 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. 2, 4

[8] 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. 1

[9] J. Lee, Jemin Hwangbo, and M. Hutter. Robust recovery controller for a quadrupedal robot using deep reinforcement learning. *ArXiv*, abs/1901.07517, 2019. 1

[10] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5, 2020. 5

[11] X. Peng, Erwin Coumans, T. Zhang, T. Lee, J. Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *Robotics: Science and Systems (RSS)*, abs/2004.00784, 2020. 3, 6, 13, 14

[12] 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. 1

[13] M. Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, K. Bodie, P. Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, A. Melzer, and M. Höpflinger. Anymal - a highly mobile and dynamic quadrupedal robot. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016. 2

[14] 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.

[15] G. Bledt, Matthew J. Powell, B. Katz, J. Carlo, P. Wensing, and Sangbae Kim. 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.

[16] 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.

[17] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan

Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016.

[18] 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. 2, 4

[19] 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. 2, 4

[20] 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.

[21] 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. 2

[22] 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. 2

[23] 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.

[24] 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. 2

[25] Nate 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. 2, 5

[26] 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.

[27] G. Endo, J. Morimoto, Takamitsu Matsubara, J. Nakanishi, and G. Cheng. Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In *AAAI*, 2005.

[28] J. Tan, T. Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *ArXiv*, abs/1804.10332, 2018.

[29] 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.

[30] Jemin Hwangbo, J. Lee, A. Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4, 2019.

[31] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5 (47):eabc5986, 2020.

[32] 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.

[33] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems (RSS)*, 2021.

[34] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. *CoRL*, 2022. 2, 5

[35] 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. 2

[36] Hae won Park, Patrick M. Wensing, and Sangbae Kim. Jumping over obstacles with mit cheetah 2. *Robotics Auton. Syst.*, 136:103703, 2021. 2

[37] Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills. *Computer Graphics Forum*, 39, 2020. 2

[38] Guillaume Bellegarda and Quan Nguyen. Robust quadruped jumping via deep reinforcement learning. *ArXiv*, abs/2011.07089, 2020. 2

[39] G. Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, Sangbae Kim, and Pulkit Agrawal. Learning to jump from pixels. In *Conference on Robot Learning*, 2021. 2, 3

[40] 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. 2

[41] Eric Vollenweider, Marko Bjelonic, Victor Klemm, N. Rudin, Joonho Lee, and Marco Hutter. Advanced skills through multiple adversarial motion priors in reinforcement learning. *ArXiv*, abs/2203.14912, 2022. 3

[42] Chenxiao Yu and Andre Rosendo. Multi-modal legged locomotion framework with automated residual reinforcement learning. *IEEE Robotics and Automation Letters*, 7:10312–10319, 2022. 3

[43] Yuni Fuchioka, Zhaoming Xie, and Michiel van de Panne. Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. *ArXiv*, abs/2210.01247, 2022. 3

[44] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Jonathan

Fiene, Alexander Badri-Spröwitz, and Ludovic Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5:3650–3657, 2019. 3

[45] Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems*, 1997. 3

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

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

[48] 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. 3

[49] 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. 3, 6

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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*, 07 2020. doi: 10.15607/RSS.2020.XVI.064. 3, 6

[55] 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. 3

[56] Alejandro Escontrela, Xue Bin Peng, Wenhao Yu, Tingnan Zhang, Atil Iscen, Ken Goldberg, and P. Abbeel. 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. 3

[57] 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. 3

[58] N. Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *ArXiv*, abs/2109.11978, 2021. 3

[59] Gabriel Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. In *Robotics: Science and Systems*, 2022. 3

[60] Atil Iscen, George Yu, Alejandro Escontrela, Deepali Jain, Jie Tan, and Ken Caluwaerts. Learning agile locomotion skills with a mentor. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2019–2025, 2020. 3, 5

[61] 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. 3

[62] 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. 3

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

[64] 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.

[65] Joshua Tobin, Rachel Fong, Alex Ray, J. Schneider, W. Zaremba, and P. Abbeel. 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.

[66] 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.

[67] 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.

[68] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[69] 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.

[70] 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. 3

[71] Jane X. Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z. Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matthew M. Botvinick. Learning to reinforcement learn. *ArXiv*, abs/1611.05763, 2017. 3

[72] 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.

[73] 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.

[74] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. Rapidly adaptable legged robots via evolutionary meta-learning. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776, 2020. 3

[75] Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2017. 3, 8

[76] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *ArXiv*, abs/1709.10087, 2017.

[77] Tim G. J. Rudner and Cong Lu. On pathologies in kl-regularized reinforcement learning from expert demonstrations. *ArXiv*, abs/2212.13936, 2022.

[78] 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. 3

[79] 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.

[80] 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, 1996. URL https://proceedings.neurips.cc/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf. 3, 8

[81] Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Manfred Otto Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *ArXiv*, abs/1707.08817, 2017. 3, 5

[82] Annie Xie and Chelsea Finn. Lifelong robotic reinforcement learning by retaining experiences. *ArXiv*, abs/2109.09180, 2021. 3

[83] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518. 4

[84] Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. 4, 5

[85] 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. 4

[86] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2022. 5

[87] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 5

[88] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. *Conference on Robot Learning (CoRL)*, abs/1910.02812, 2018. 5

[89] Tsung-Yen Yang, Tingnan Zhang, Linda Luu, Sehoon Ha, Jie Tan, and Wenhao Yu. Safe reinforcement learning for legged locomotion. *ArXiv*, abs/2203.02638, 2022. 5

[90] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016. 5

[91] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 5

[92] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *ArXiv*, abs/2208.07860, 2022. 5

[93] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. 5

[94] 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.

[95] Alexander Matt Turner, Neale Ratzlaff, and Prasad Tadepalli. Avoiding side effects in complex environments. *arXiv preprint arXiv:2006.06547*, 2020. 5

[96] 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. 6

Appendix

A. *Experimental Details*

1) *Further Environment Details:*

a) 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.

b) 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 $= 4.0$.

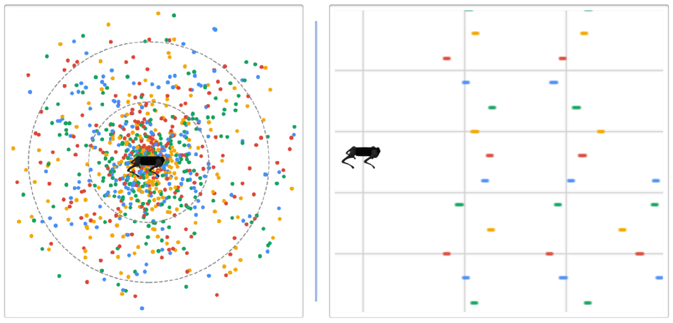


Fig. 10: Distribution of goals in the two tasks we consider. (Left) We visualize the bipedal navigation task, where each colored dot is a sample from the goal distribution we defined. The small dotted circle corresponds to a radius of 1m and the larger circle corresponds to a radius of 2m.

c) 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 [11]. 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 II). 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 [11], where the reward r_t at each timestep is calculated according to:

$$r_t = \sum_i w^i r_t^i \tag{3}$$

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

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). 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 3 with w's and r's listed in Table II.

d) 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 $\mathrm{v}_{\text{forward}}$ by rotating $[1, 0, 0]$ by the robot's orientation. We then compute the cosine distance

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

Lastly, we normalize to be between 0 and 1:

$$r^{\text{upright}} = (0.5 \cdot \texttt{cos\_dist} + 0.5)^2. \tag{4}$$

To encourage the robot to lift itself up, we define

$$r^{\text{max height}} = \exp(\texttt{root\_height}) - 1, \tag{5}$$

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

$$r^{\text{stand}} = r^{\text{upright}} + r^{\text{max height}} \tag{6}$$

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

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 I: Reward terms encouraging tracking a jumping reference motion, from [11].

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	$\text{count}\left[\text{hurdles passed}\right]$	0.1	Bonus for clearing each hurdle

Table II: Exact definition of each of the reward terms used in our overall jumping reward function defined in Equation 3.

robot to walk to a wide variety of goal locations (see left of Figure 10). The reward function is a combination of the standing reward and a simple forward locomotion reward. Specifically, we define the reward to be

$$r_t^{\text{stand}} \cdot (1 + r_t^{\text{facing}} + r_t^{\text{distance}}). \tag{7}$$

where r_t^{facing} follows Equation 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.

e) 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.

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 III.

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

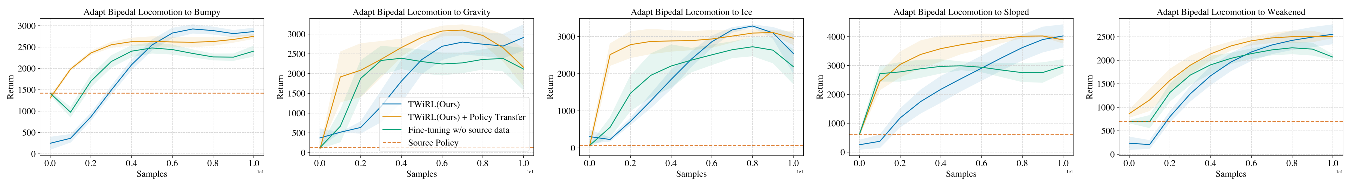
Table III: Hyperparameters shared for training source policies with DroQ and for transferring with Algorithm 1.

B. Detailed Experiments

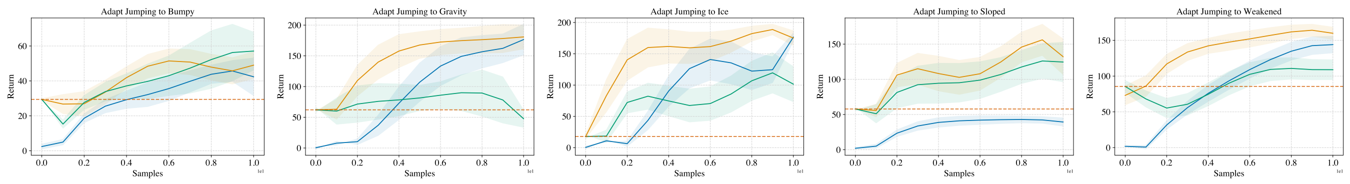
1) Full results: We present all learning curves for evaluating TWiRL's ability to facilitate transfer to different environments in Figure 11. We see that in every case except for adapting the jumping policy to the sloped environment, vanilla TWiRL successfully learns a policy that works in the new environment. 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.

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.

a) 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 procedure defined in Algorithm 1. 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 12. 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.



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



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

Fig. 11: Individual learning curves for every task and target environment combination.

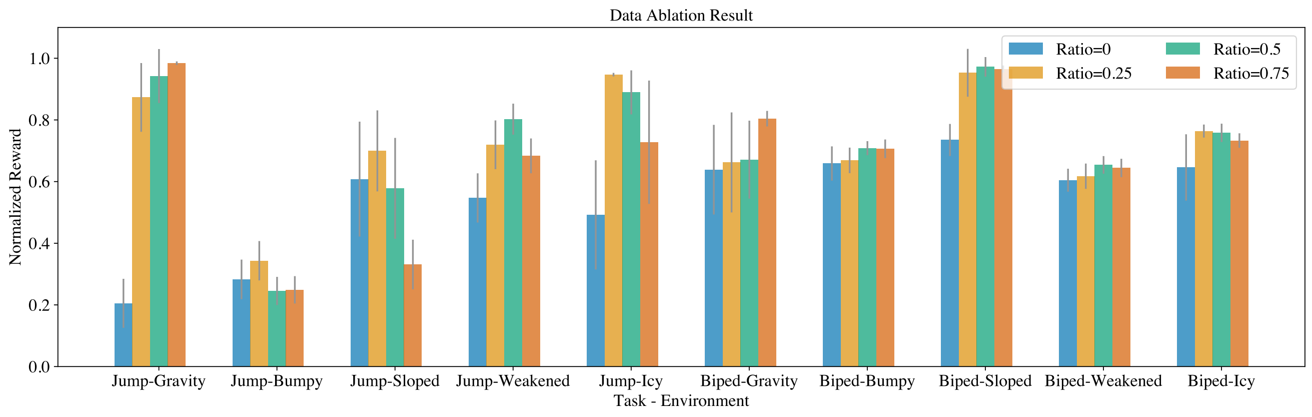


Fig. 12: Studying the effect of using data from π_{src}.

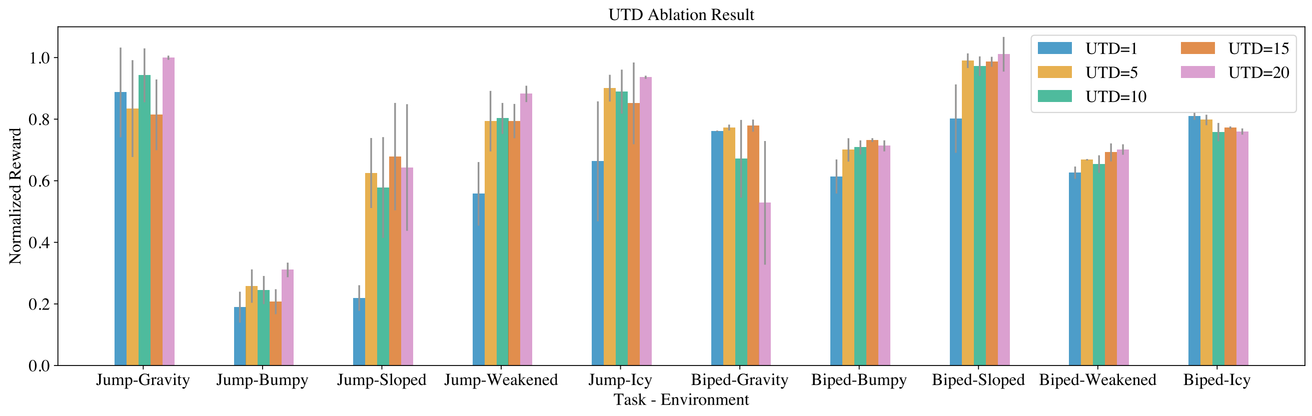


Fig. 13: Studying the effect of using high UTD algorithms for incorporating off-policy data. We find that using an UTD ratio ¿ 1 is important for stable learning particularly when learning the jumping policies in a new environment.