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

Grow Your Limits: Continuous Improvement with Real-World RL for Robotic Locomotion

Laura Smith*[1], Yunhao Cao*[1], Sergey Levine[1]
*Equal contribution [1]Berkeley AI Research, UC Berkeley
{smithlaura, caoyh2001}@berkeley.edu

***Abstract*— Deep reinforcement learning can enable robots to autonomously acquire complex behaviors such as legged locomotion. However, RL in the real world is complicated by constraints on efficiency, safety, and overall training stability, which limits its practical applicability. We present APRL, a policy regularization framework that modulates the robot's exploration throughout training, striking a balance between flexible improvement potential and focused, efficient exploration. APRL enables a quadrupedal robot to efficiently learn to walk entirely in the real world within minutes and continue to improve with more training where prior work saturates in performance. We demonstrate that continued training with APRL results in a policy that is substantially more capable of navigating challenging situations and adapts to changes in dynamics. Videos and code to reproduce our results are available at: https://sites.google.com/berkeley.edu/aprl**

I. Introduction

Confronting the noisy, diverse, and unpredictable nature of the real world demands a high degree of robustness and versatility from robotic systems. Designing controllers that anticipate and handle *any* scenario a robot will encounter in its lifetime, whether through manual engineering or learning, is impractical. Legged robots in particular have incredible mobility—they can reach territory unsuitable even for humans, for example, in search and rescue situations. Instead of providing it with predetermined capabilities, we seek to equip the robot with the ability to adapt on its own, on the spot, to such unanticipated circumstances.

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 [1], [2], 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 [3]–[10] 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 work, 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 scenarios?

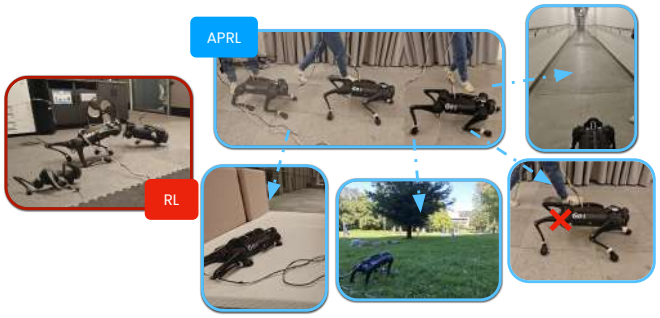

Fig. 1: APRL uses a novel action space regularization technique based on dynamics prediction error to modulate exploration over the course of training. This enables real-world quadrupedal learning that can traverse challenging terrains and continually adapt to changes in dynamics.

We present APRL, a system that addresses the real-world challenges of *efficiency* and *continual improvement* in robot learning via adaptive policy regularization, focusing on quadrupedal locomotion. Our key observation is that the policy's search space of actions significantly affects the robot's learning capacity. To illustrate this, consider tasking the robot to learn to walk without 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 1). But manually defining an appropriate space of actions to explore restricts the robot's capabilities, as peak performance 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 learn this through costly first-hand experience. However, 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 robot is in its current situation. This 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 expect the policy to generalize poorly. 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 [11], which combats early overfitting, a common problem sample-efficient RL algorithms are prone to. Resets

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

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, the main contribution of our work is 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 a 12 degree-of-freedom Unitree Go1 quadruped can learn to walk forward in just minutes starting completely from scratch, and while prior work saturates in performance, APRL allows the robot to continuously improve with time. 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 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 results is available on our website: `https://sites.google.com/berkeley.edu/aprl`

II. Related Work

Legged locomotion has long been of interest to roboticists. A large body of work is dedicated to developing controllers by means of model-based optimal control [12]–[18], and these have enabled a range of robotic locomotion skills, from high-speed running [19] to backflipping [20] but require careful modeling of the conditions in which they will perform. Learned approaches have recently seen remarkable success, largely by leveraging simulation to train robust behaviors—demonstrating impressive transferability to the real world from walking [21]–[32] to more agile behaviors like running [33], jumping [34], [35] or bipedalism [36]–[39]. While this approach has been sufficient in many scenarios of interest, relying on zero-shot policy generalization has two key limitations: it requires extensive engineering of simulated settings, and the robot has no recourse when its policy fails.

We instead study learning directly in the environment of interest. Early work in this vein explored utilizing higher level actions in trajectory space [3], [40]–[44], 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. [45] report learning to walk without resets and to recover from pushes in 1 hour using model-based RL. We build our system on the simple model-free framework [46] that demonstrated learning to walk from scratch in various environments, each within 20 minutes [46]. 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 use model prediction error to regularize actions, which leads to fewer falls and lower torques during training. For locomotion, falls lead to physical damage and increased wall-clock training time. To address this, for example, Ha et al. [47] use a constrained MDP (CMDP) formulation to explicitly combat falling during training by penalizing actions that lead to fallen states. In Section VI, we find empirically that the CMDP approach still requires many failures to do effective credit assignment. Our action regularization is similar insofar as it manifests as a penalty in the actor's objective, but we directly regulate the magnitude of actions the policy explores within. Therefore, in APRL, the policy does not first need to learn to avoid dangerous actions by taking them and experiencing failure instead we leverage our knowledge that small actions around the nominal pose are relatively safe, 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.

III. 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 build on the actor-critic RL method and implementation by Smith et al. [46] (details in *(b)*), which fits a critic $Q_\theta(\mathbf{s}, \mathbf{a})$ to estimate the above quantity. The policy then uses the critic to improve by maximizing the estimated return:

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

We additionally use resets to improve plasticity [11], [48]. All neural networks are 2-layer feed-forward networks constructed and trained using JAX [49]. We use an Origin EON15-X laptop with an NVIDIA GeForce RTX 2070 GPU.

a) MDP formulation: We use the Go1 quadruped from Unitree for real-world experiments and perform analysis in simulation using the MuJoCo Menagerie model [50]. 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, joint velocities, root (local) velocity, normalized foot contacts, and last recorded action. Previous works [45], [46] used a Kalman filter to fuse forward kinematics and acceleration to estimate the robot's velocity. In this work, we instead obtain velocity estimates

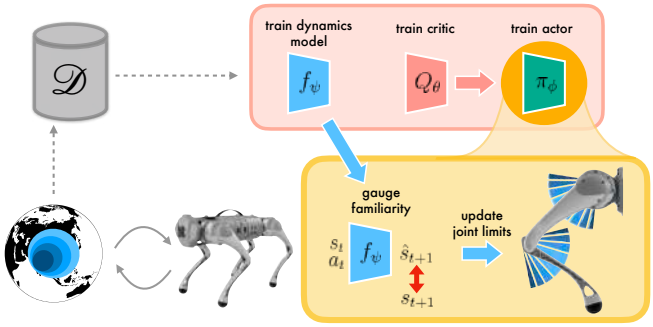

Fig. 2: **Overview of APRL.** We train the robot with a flexible constraint, represented by blue circular sectors around the joints. The robot collects experience, storing it in a replay buffer for training an actor and critic, as explained in Section III, alongside a predictive dynamics model. The model's 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 objective, as specified in Equation 1.

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. The robot learns to output target joint positions at a rate of 20 Hz. These targets are fed to a PD controller, with a position gain of 20 and derivative gain of 1, which converts them to torques. We employ two smoothing mechanisms: a low-pass filter in the policy output space and action interpolation in the PD controller (running at 500 Hz). We define a reward function to maximize the robot's local linear velocity while maintaining an upright orientation and include penalties on angular velocity and torque smoothness. The definition of all reward terms can be found on our website.

b) Replay ratio and resets: Many sample-efficient RL methods use a high *replay ratio*, i.e., the ratio of gradient updates to real-world transitions collected, to use the collected data efficiently. To be able to take more updates on the same data, they require some form of regularization, e.g., using model-generated data [51], [52], weight-based regularization [53], [54], ensembling at the agent level [55] or at the critic level [56], or a combination of techniques [57]. In this work, we use a high replay ratio with a model-free method, using Dropout [58] and LayerNorm [59] to regularize the critic. Nikishin et al. [11] 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 replay buffer. We incorporate this regularizer into our adaptive strategy as we will describe next.

IV. 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). As shown in Figure 2, our framework dynamically modulates regularization as the robot trains to provide it with adequate room to explore and improve without suffering unnecessarily 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 detail how we adapt the constraints based on the robot's learning progress and finally how we implement them in practice. Algorithm 1 summarizes the training procedure in pseudocode.

Algorithm 1 `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 $i \leftarrow i + 1$
  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 $\mathcal{L}^{\texttt{dyn}}_{\texttt{act}}(\psi)$ in Equation 2
  Update ϕ with $\mathcal{L}^{\texttt{APRL}}_{\texttt{act}}(\phi)$ in Equation 1
  // Periodically reset weights
  if $i \cdot rr > \nabla_M$ **then**
    Reinitialize θ, ϕ, ψ and reset $i = 0$
until forever

a) An efficiency-performance trade-off: Prior work has manually defined explicit ranges for joint position actions, especially for high-dimensional robots to avoid dangerous behavior like self-collisions [60], [61]. Smith et al. [46] further showed that this design factor has an enormous effect on learning efficiency in the real world. Intuitively, limiting the policy's search space makes finding a solution faster. Furthermore, legged locomotion policies are often parameterized to output PD targets as actions [29], [32], [39], [62], so 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 makes learning to walk remarkably efficient, it can significantly inhibit the learned policy's ultimate capabilities.

b) Soft policy constraints: A straightforward approach to constraining the policy's actions within fixed bounds is to use a transformation that either maps any action beyond the bounds to the corresponding extremum or bounds and

scales the policy network to output actions within the defined limits. We find that these naïve implementations do not work well in practice (see Section VI) 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 element-wise 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 objective:

$$\mathcal{L}^{\texttt{APRL}}_{\texttt{act}}(\phi) = \mathop{\mathbb{E}}_{\substack{s\sim D \\ a\sim\pi_\phi(\cdot|s)}} \Big[Q_\theta(\mathbf{s}, \mathbf{a}) - \sigma \sum_i \max(0, c_i(\mathbf{a})) \Big]. \tag{1}$$

c) Deciding on appropriate feasible regions: We design our method such that the robot can be more exploratory when in a familiar setting and, conversely, more conservative in settings different from those it has already trained on. 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'. While there are several ways to capture a notion of familiarity, e.g., using model disagreement to measure epistemic uncertainty [63], we use misprediction for a few reasons. First, it implicitly encourages the policy to favor predictable behaviors. Secondly, it 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:

$$\mathcal{L}^{\texttt{dyn}}(\psi) = \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},\mathbf{s}_{t+1})\sim D} \Big[\frac{1}{2} \| (\hat{f}_\psi(\mathbf{s}, \mathbf{a}) - \mathbf{s}_{t+1} \|^2 \Big]. \tag{2}$$

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. The exact hyperparameters used in our experiments can be found on our website.

V. 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 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?

Fig. 3: **Environment visualization.** We visualize the different environments we test in: Grass, Ramp, Mattress, and Frozen Joint. For the environments where there is an explicit path for the robot to traverse that we evaluate on, we indicate the start and goal locations with white circles and gold stars.

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?

A. Experimental Setup

We address the first two questions by comparing to the prior work of Smith et al. [46] (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 3):

- **Mattress**: The robot must walk across a 5 cm 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.
- **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 in terrain to cause differences in performance.

To account for stochasticity and variance in the real world, we run each evaluation three 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 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 that we do not include the relative finish time and fall counts for the *Frozen Joint* scenario because the dynamics shift is not in a terrain that the robot can traverse for us to record these metrics. Lastly, for (4), we evaluate whether a few minutes of continued training (specifically 3000 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.

Fig. 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.

Fig. 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 traverse it more efficiently, and after fine-tuning, with fewer falls.

B. Results

Training from scratch. In Figure 6, we see that even without manually restricting the actions the robot can take, APRL starts learning to walk immediately due to the immediate influence of the regularization imposed on the actor. We attempted to compare to training without regularization; however, the robot's actions were too aggressive to collect even a few thousand transitions. APRL's adaptive regularization makes training possible in a 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, ultimately achieving a maximum average velocity of only 0.44 m/s. This performance almost matches its simulated counterpart (see Section VI), so we believe this cap to be a fundamental limitation rather than a challenge specific to the real world. 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 4 and better viewed in video form on our project website. These results show that APRL is significantly better equipped than naïve RL to continually improve as it collects more data, as opposed to quickly reaching but plateauing with limited capabilities.

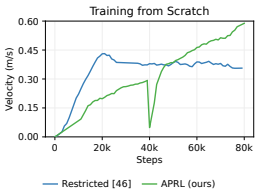

Fig. 6: **Forward velocity achieved during training.** The *Restricted* method learns more efficiently at first but is not able to keep improving. Meanwhile, our method still learns to walk quickly but acquires a maximum velocity of 0.62 m/s. Note that the dip at 4k steps is due to resets to improve plasticity (line 19 in Algorithm 1).

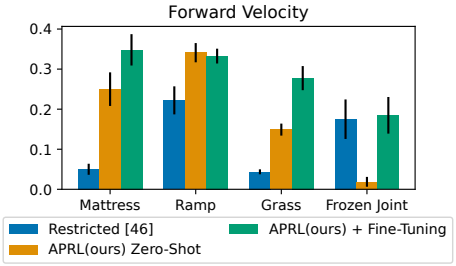

Fig. 7: **Real-world velocity comparisons (higher is better).** In all scenarios except *Frozen Joint*, APRL significantly outperforms the *Restricted* method in terms of its velocity when tested in new scenarios. With just minutes of fine-tuning, APRL significantly improves performance in all settings except on the *Ramp*, where it is comparable.

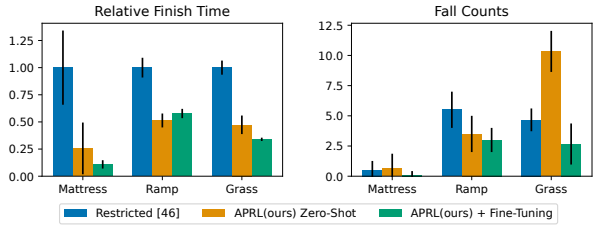

Fig. 8: **Real-world finish time and fall count (lower is better).** (Left) We report the time taken to traverse a path relative to the *Restricted* method and absolute fall count. On *Ramp* and *Grass*, APRL is **2x** faster, and on *Mattress*, APRL is **almost 4x** faster.

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 7) and at completing a given path faster and with fewer falls (see Figure 8). The exception is when we freeze a joint—in this scenario, the *Restricted* policy exhibits much better zero-shot generalization. In this case, we find that with continued training, APRL can quickly learn to overcome this gap. In Figure 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 on our project website.

VI. Simulated Analysis

In this section, we analyze APRL using a simulated version of the task described in Section III. Although simulation does not capture many of the real-world challenges 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?
2) If so, does APRL 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 [46]

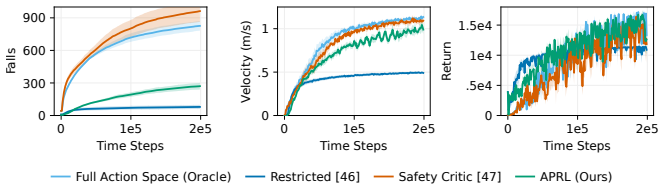

Fig. 9: **Comparisons.** We report each policy's performance measured by the falls, average velocity, and return (mean and standard error across 5 seeds) with respect to the number of time steps. We find that APRL is the only method that effectively balances achieving high velocity while regulating the number of falls such that it is feasible to run in the real world.

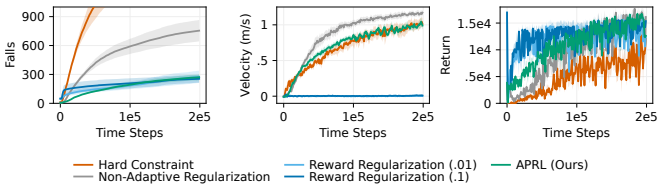

Fig. 10: **Ablations.** We compare to versions of APRL that use (a) a hard constraint (b) non-adaptive regularization (c) regularization via the reward function. These either have too many falls or do not progress on *forward velocity*, showing the importance of all design components of APRL.

(labeled *Restricted*) to learning with the full action range without our adaptive regularization (labeled *No Regularization*). As mentioned in Section V, 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 9 that the way actions are explored profoundly affects 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 does not improve beyond a velocity of about 0.5 m/s. In contrast, the policies with access to the full joint range eventually far exceed the *Restricted* policy's performance.

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. [47] 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 the safety critic required tens of thousands of samples to converge in our application, which was not quick enough to shape the early stages of exploration to prevent excessive falls. In fact, this method was especially sensitive to network initialization, 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 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 1) 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. 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.

VII. Conclusion

We presented APRL, 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.

Acknowledgements

This work was supported in part by ARO W911NF-21-1-0097, ARL DCIST CRA W911NF-17-2-0181, and ONR N00014-20-1-2383 and N00014-22-1-2773. Laura Smith is supported by a Google PhD Fellowship.

References

[1] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," *ArXiv*, vol. abs/2004.12570, 2020. 1

[2] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, pp. 698 – 721, 2021. 1

[3] 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*, vol. 3, pp. 2619–2624 Vol.3, 2004. 1, 2

[4] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793, 2017. 1

[5] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *Conference on Robot Learning (CoRL)*, vol. abs/1806.10293, 2018. 1

[6] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, pp. 421 – 436, 2018. 1

[7] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *ArXiv*, vol. abs/2104.08212, 2021. 1

[8] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, "Bridge data: Boosting generalization of robotic skills with cross-domain datasets," *ArXiv*, vol. abs/2109.13396, 2022. 1

[9] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2019. 1

[10] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," *Conference on Robot Learning (CoRL)*, vol. abs/1909.11652, 2019. 1

[11] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. C. Courville, "The primacy bias in deep reinforcement learning," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248811264 1, 2, 3

[12] M. Kalakrishnan, J. Buchli, P. Pastor, M. N. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," *2010 IEEE International Conference on Robotics and Automation*, pp. 2665–2670, 2010. 2

[13] D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, pp. 2261–2268, 2018. 2

[14] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302. 2

[15] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, pp. 429–455, 2016. 2

[16] M. Hutter, C. Gehring, D. Jud, A. Lauber, D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Höpflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44, 2016. 2

[17] G. Bledt, M. J. Powell, B. Katz, J. Carlo, P. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2245–2252, 2018. 2

[18] B. Katz, J. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6295–6301, 2019. 2

[19] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the mit cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017. [Online]. Available: https://doi.org/10.1177/0278364917694244 2

[20] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H. won Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, pp. 1154–1171, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:229331611 2

[21] J. Tan, Z. Xie, B. Boots, and C. Liu, "Simulation-based design of dynamic controllers for humanoid balancing," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2729–2736, 2016. 2

[22] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *ArXiv*, vol. abs/1804.10332, 2018. 2

[23] Z. He, R. C. Julian, E. Heiden, H. Zhang, S. Schaal, J. J. Lim, G. Sukhatme, and K. Hausman, "Zero-shot skill composition and simulation-to-real transfer by learning task representations," *ArXiv*, vol. abs/1810.02422, 2018. 2

[24] W. Yu, V. Kumar, G. Turk, and C. Liu, "Sim-to-real transfer for biped locomotion," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3503–3510, 2019. 2

[25] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," *Conference on Robot Learning (CoRL)*, vol. abs/1910.02812, 2018. 2

[26] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. V. D. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning (CoRL)*, 2019. 2

[27] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, 2019. 2

[28] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, 2020. 2

[29] X. Peng, E. Coumans, T. Zhang, T. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *Robotics: Science and Systems (RSS)*, vol. abs/2004.00784, 2020. 2, 3

[30] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," *ArXiv*, vol. abs/2104.04644, 2021. 2

[31] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *ArXiv*, vol. abs/2109.11978, 2021. 2

[32] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *Robotics: Science and Systems (RSS)*, 2021. 2, 3

[33] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022. 2

[34] N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Transactions on Robotics*, vol. 38, pp. 317–328, 2021. 2

[35] G. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. Kim, and P. Agrawal, "Learning to jump from pixels," in *Conference on Robot Learning*, 2021. 2

[36] C. Yu and A. Rosendo, "Multi-modal legged locomotion framework with automated residual reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, pp. 10 312–10 319, 2022. 2

[37] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," *ArXiv*, vol. abs/2203.14912, 2022. 2

[38] Y. Fuchioka, Z. Xie, and M. van de Panne, "Opt-mimic: Imit@articleAchiam2017ConstrainedPO, title=Constrained Policy Optimization, author=Joshua Achiam and David Held and Aviv Tamar and P. Abbeel, journal=ArXiv, year=2017, volume=abs/1705.10528, url=https://api.semanticscholar.org/CorpusID:10647707 ation of optimized trajectories for dynamic quadruped behaviors," *ArXiv*, vol. abs/2210.01247, 2022. 2

[39] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Learning and adapting agile locomotion skills by transferring experience," *Robotics: Science and Systems (RSS)*, 2023. 2, 3

[40] R. 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)*, vol. 3, pp. 2849–2854 vol.3, 2004. 2

[41] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid," in *AAAI*, 2005. 2

[42] K. S. Luck, J. Campbell, M. A. Jansen, D. M. Aukes, and H. B. Amor, "From the lab to the desert: Fast prototyping and learning of robot locomotion," *Robotics: Science and Systems (RSS)*, vol. abs/1706.01977, 2017. 2

[43] S. Choi and J. Kim, "Trajectory-based probabilistic policy gradient for learning locomotion behaviors," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2019. 2

[44] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," *Conference on Robot Learning (CoRL)*, vol. abs/1907.03613, 2019. 2

[45] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Daydreamer: World models for physical robot learning," *Conference on Robot Learning (CoRL)*, vol. abs/2206.14176, 2022. 2

[46] L. Smith, I. Kostrikov, and S. Levine, "Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," *Robotics: Science and Systems (RSS) Demo*, 2023. 2, 3, 4, 5

[47] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *ArXiv*, vol. abs/2002.08550, 2020. 2, 6

[48] P. D'Oro, M. Schwarzer, E. Nikishin, P.-L. Bacon, M. G. Bellemare, and A. C. Courville, "Sample-efficient reinforcement learning by breaking the replay ratio barrier," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259298604 2

[49] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax 2

[50] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022. [Online]. Available: http://github.com/deepmind/mujoco_menagerie 2

[51] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *ArXiv*, vol. abs/1906.08253, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:195068981 3

[52] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *International Conference on Learning Representations (ICLR)*, 2020. 3

[53] Z. Liu, X. Li, B. Kang, and T. Darrell, "Regularization matters in policy optimization - an empirical study on continuous control," *arXiv: Learning*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:222176807 3

[54] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka, "Dropout q-functions for doubly efficient reinforcement learning," *International Conference on Learning Representations (ICLR)*, 2022. 3

[55] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning," *ArXiv*, vol. abs/2007.04938, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:220424803 3

[56] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *ArXiv*, vol. abs/2101.05982, 2021. 3

[57] Q. Li, A. Kumar, I. Kostrikov, and S. Levine, "Efficient deep reinforcement learning requires regulating overfitting," *ArXiv*, vol. abs/2304.10466, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258236460 3

[58] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014. 3

[59] J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv*, vol. abs/1607.06450, 2016. 3

[60] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," *Conference on Robot Learning (CoRL)*, 2021. 3

[61] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. M. O. Heess, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *ArXiv*, vol. abs/2304.13653, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258331581 3

[62] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," *Conference on Robot Learning (CoRL)*, 2021. 3

[63] P. Shyam, W. Jaśkowski, and F. J. Gomez, "Model-based active exploration," in *International Conference on Machine Learning*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:53102049 4