2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning

Xingyou Song*† Yuxiang Yang*†§ Krzysztof Choromanski†
Ken Caluwaerts† Wenbo Gao‡ Chelsea Finn† Jie Tan†
†Robotics at Google ‡Columbia University

***Abstract*—Learning adaptable policies is crucial for robots to operate autonomously in our complex and quickly changing world. In this work, we present a new meta-learning method that allows robots to quickly adapt to changes in dynamics. In contrast to gradient-based meta-learning algorithms that rely on second-order gradient estimation, we introduce a more noise-tolerant Batch Hill-Climbing adaptation operator and combine it with meta-learning based on evolutionary strategies. Our method significantly improves adaptation to changes in dynamics in high noise settings, which are common in robotics applications. We validate our approach on a quadruped robot that learns to walk while subject to changes in dynamics. We observe that our method significantly outperforms prior gradient-based approaches, enabling the robot to adapt its policy to changes based on less than 3 minutes of real data.**

I. Introduction

Deep reinforcement learning (RL) holds the promise of automatically acquiring locomotion controllers for legged robots [14, 17, 43, 47]. Typically, these methods train a control policy in a computationally efficient simulation environment and then deploy the learned policy to hardware (i.e. sim-to-real). However, the policies learned with deep RL often only work well in environments highly similar to those they were trained on. Any mismatch or non-trivial changes to the environment can require re-training from scratch. For robots to operate autonomously in our complex and ever-changing world, it is crucial that they can adapt their control policies quickly to accommodate these changes. For example, a legged robot may need to change its locomotion gait if its battery level is low, if a motor is broken, or if it is carrying a heavy load.

Meta-learning leverages previous experience to explicitly train a policy to be adaptable, by training across many different tasks or environment conditions. One such approach specifically designed to make policies adaptable is *model agnostic meta-learning* (MAML) [9]; MAML performs adaptation via gradient updates on policy parameters using a small amount of data, optimizing so that such updates are fast and efficient. Since then, numerous works have improved upon this paradigm [41, 48], usually using simulators such as MuJoCo [45] and PyBullet [4] for evaluating and comparing algorithms.

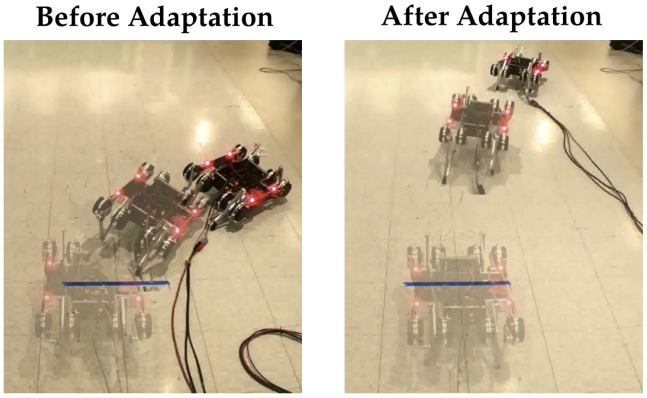


Fig. 1: Our algorithm quickly adapts a legged robot's policy to dynamics changes. In this example, the battery voltage dropped from 16.8V to 10V which reduced motor power, and a 500g mass was also placed on the robot's side, causing it to turn rather than walk straight. Based on only 50 episodes (150s) of real world data, our method adapted the policy to significantly improve performance.

However, applying these techniques to real robots is challenging due to the noise present in real world environments. Even with the same control policy, the trajectories of different runs can quickly diverge and lead to significantly different reward signals. This stochasticity renders gradient estimates unreliable during adaptation, unless we dramatically increase the amount of data collected. Different initial conditions, hardware wear-and-tear, observation noise, and action noise make exact repetition of trials difficult. This problem is especially evident for legged locomotion for which contact events are frequent and a small difference in contacts can generate significantly different trajectories.

The primary motivation for this work is: How we can ensure effective adaptation in *highly noisy environments*? When using the class of policy gradient algorithms [42, 38, 37] which require an action *distribution*, we may at first glance wish to *decrease* the entropy of this distribution as the high-noise problem might be exacerbated by the additional randomness from the policy's actions. However, we may

*Equal contribution.

§Work performed during the Google AI Residency Program. http://g.co/airesidency

Video can be found at https://youtu.be/_QPMCDdFC3E.

Code can be found at https://github.com/google-research/google-research/tree/master/es_maml.

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

also wish to *increase* this entropy as well, as the policy needs random actions for *exploration* to determine the type of environment to adapt to. These two competing objectives, which seek to both decrease and increase action entropy, may cause complications when using policy gradients, as noted in [39, 23].

The class of evolutionary strategies (ES) [36, 25] algorithms are attractive approaches to rectify these problems. ES leverages parameter-space search, which allows it to learn deterministic policies that are particularly conducive to adaptation. In this work, we combine ES with meta-learning, adopting the ES-MAML framework [41]. This framework allows us to explore a more powerful and non-differential adaptation process. To combat stochasticity in noisy real-world environments, we introduce a noise-tolerant and data-efficient adaptation operator based on Batch Hill-Climbing (BHC). Since this operator is not differentiable, it is difficult to integrate with the state-of-the-art policy-gradient-based MAML paradigms (PG-MAML) [9]. Hence, we use ES-MAML, and show that ES-MAML with BHC is not only empirically more effective than MAML, but also theoretically justifiable under conditions with strong noise.

Furthermore, ES-MAML allows adaptation using compact linear policies, which have been shown to qualitatively produce more stable behaviors [25, 32]. While policy gradients are also able to train linear policies [32], effective adaptation with PG-MAML relies on flexible representational capacity [8]; thus, linear policies and meta-learning may not combine well for policy gradients.

We validate our approach on the Minitaur, a quadruped robot platform. We demonstrate that a policy trained purely in simulation using our method can not only overcome the sim-to-real gap, but also adapts successfully to two difficult new real-world environments: 1) a low battery level with extra payload and 2) a slippery walking surface. We further compare our method with PG-MAML approaches and conduct ablation studies on various design choices of our method. We show that our method adapts faster and achieves high rewards post-adaptation, even in environments that are out of distribution from training. We also show that our BHC method is more noise tolerant than other Hill-Climbing methods, which is essential for real-world experiments. In summary, the contributions of this paper include:

1) We introduce and integrate BHC, a non-differentiable, highly-efficient and noise-tolerant adaptation operator, into ES-MAML to learn adaptable locomotion policies for legged robots.
2) We successfully demonstrate fast adaptation on a quadruped robot. It requires only 50 episodes totalling 150 seconds of robot data, to adapt the control policies in two drastically different environments.
3) We provide a thorough evaluation of ES-MAML with BHC both in simulation and on a real robot, which demonstrates its advantage over gradient-based meta-learning techniques.

II. Related Work

A. Learning Locomotion

Reinforcement learning is becoming a popular method to develop locomotion controllers for legged robots [44, 21, 47, 40, 20, 19, 43, 17]. Learning can be performed in simulation and followed by a sim-to-real transfer phase [43, 17] or carried out directly on real hardware [14, 13]. Since legged robots can walk in diverse environments, the ability to change their learned policies to fit new environments is important. Prior work has investigated online system identification [49], action transformation [15], precomputed behavior-performance map [5], strategy optimization [50, 51] and meta-learning [29] for quick adaptation of locomotion controllers.

B. Policy Transfer

Transferring a learned policy to different dynamics has been a long-standing problem in reinforcement learning. *Domain Randomization (DR)* trains the policy with experiences collected from multiple dynamics, so that the learned policy can remain robust across a number of environments [43, 31, 35]. More recently, advanced ways of domain randomization have been proposed such as hierarchical domain randomization [28], curriculum sampling [26], and inverse dynamics [3]. Although DR may work effectively for sim-to-real transfer, it trades optimality for robustness and makes the learning problem much harder.

Another class of approaches augment the state space with a latent vector, which represents the environment changes. The latent vector can be based on actual physics parameters [49, 50] or based on a learned embedding of the trajectory [33]. During adaptation, the algorithm either infers the latent representation of the environment through system identification [49], or directly optimizes the latent variable for optimal performance [51, 50]. While this approach has demonstrated adaptation in several real-robot scenarios [50, 51], choosing the parameters of the latent representation requires careful tuning.

C. Meta-Learning

More relevant to our work, meta-learning [9] has been used for policy transfer in reinforcement learning where the problem was defined as finding a *meta-policy* that can be adapted to a new task in a few gradient steps. The original Model-Agnostic Meta-Learning (MAML) algorithm [9] uses policy gradient methods as the policy adaptation operator, which we refer to as *PG-MAML*. Several improvements of PG-MAML have been proposed: stochastic gradient adaptation [10], improving estimation of the Hessian [24, 34], or replacing second derivatives with multiple gradient steps (Reptile, [30]). Adaptation can be performed explicitly on a latent space [12, 1, 52], or learned via RNN-based methods [6, 46, 27]. However, most of these methods are so far only tested in basic simulated environments.

In order to leverage MAML for dynamics adaptation in real world environments, [29] combines *model-based methods*

with MAML, and presents results on a small hexpod robot. Arndt et al. [1] is one of the few works to date that has applied the model-free PG-MAML to train a robotic arm to hit a hockey puck. In contrast, our locomotion environment is significantly noisier [43] due to frequent and discrete contact events inherent to locomotion, as well as high uncertainties about the robot's dynamics due to cheap hardware.

III. Preliminaries

A. Problem Setup

We formulate the meta reinforcement learning problem as an optimization over a distribution of tasks: $T_i \in P(\mathcal{T})$, where each task is a Markov Decision Process (MDP). In our setting, all tasks share the same state space $\mathcal{S}$, the same action space $\mathcal{A}$, and most importantly *the same reward function* $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. However, given a fixed state s_t, each task can have a different transition distribution $p(s_{t+1}|s_t, a_t)$ which corresponds to different dynamics. We parameterize a policy $\pi : \mathcal{S} \to \mathcal{A}$ as a neural network with parameters $\theta \in \Theta$, denoted as π_θ.

We seek to find a policy that can adapt to different tasks efficiently. Formally, given an adaptation procedure $U : \Theta \times \mathcal{T} \to \Theta$, we optimize for a meta-policy $\pi_{\theta_{meta}}$ that maximizes the expected return of the adapted policies:

$$\theta_{meta} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{T\sim\mathcal{P}(\mathcal{T})}\left[f^T(U(\theta, T))\right] \tag{1}$$

where $f^T(\theta) = \mathbb{E}_{(s_{0:H}, a_{1:H})\sim\pi_\theta, T}\left[\sum_{t=1}^{H} r(s_t, a_t)\right]$ is the expected total sum of rewards using policy π_θ over the dynamics corresponding to task T. In the MAML algorithm [10], the update operator U corresponds to one or a few steps of gradient descent, and maximization of the objective in Eq. 1 is performed with gradient-based optimization. We define the *adaptation gap* between the meta-policy θ_{meta} and the adapted policy $U(\theta_{meta}, T)$ as the difference $f^T(U(\theta_{meta}, T)) - f^T(\theta_{meta})$, or also its expectation across $\mathcal{P}(\mathcal{T})$ when multiple tasks are involved, depending on context.

B. Meta-Learning with Evolutionary Strategies

Previous works [7, 16] replace some components of the MAML algorithm such as the outer-loop with evolutionary methods, but still rely on policy gradients to train differentiable stochastic policies. ES-MAML [41] is an algorithm under the meta-learning paradigm which uses techniques for optimizing *blackbox* objectives.

In our setting, we use a low variance *antithetic* estimator [25] for the outer loop of the ES-MAML algorithm [41]. This estimates the gradient $\nabla_\theta \mathbb{E}_{\mathbf{g}\sim\mathcal{N}(0,\mathbf{I})}[F(\theta+\sigma\mathbf{g})] = \frac{1}{2\sigma}\mathbb{E}[f(\theta+\sigma\mathbf{g}) - f(\theta-\sigma\mathbf{g})]$ of the Gaussian-smoothed version of the outer loop objective $F(\theta) = \mathbb{E}_{T\sim\mathcal{P}(\mathcal{T})}[f^T(U(\theta, T))]$. We present this algorithm formally in Algorithm 1.

We use the iterative Hill-Climbing (HC) operator U_{HC} in [41], which has been shown to be a strong adaptation operator for classic MAML benchmarks. HC has multiple benefits: for example in the deterministic case, it enforces monotonic improvement in the objective. More formally, given an adaptation rate α and a query number (Q) for number of rollouts allowed during finetuning, the Hill-Climbing step is defined as simply $\theta^{(q+1)} = \operatorname{argmax}_{\theta\in\{\theta^{(q)}, \theta^{(q)}+\alpha\mathbf{g}\}} f(\theta)$, for which we apply iteratively $\theta_{meta} \to \theta^{(1)} \to ... \to \theta^{(Q)}$ to produce a Hill-Climbing operator $U_{HC}(\theta_{meta}) = \theta^{(Q)}$. Note that U_{HC} is a non-differentiable operator which cannot be replicated in the PG case, as the Hill-Climbing operator uses argmax.

Algorithm 1 ES-MAML (Antithetic Variant) using general adaptation operator $U(\cdot, T)$.

Data: initialized meta-policy θ_{meta}, meta step size β
while *not done* **do**
  Sample n tasks $T_1, \ldots, T_n \sim \mathcal{P}(\mathcal{T})$ and i.i.d. vectors $\mathbf{g}_1, \ldots, \mathbf{g}_n \sim \mathcal{N}(0, \mathbf{I})$ **foreach** $(T_i, \mathbf{g}_i)$ **do**
    $v_i^+ \leftarrow f^{T_i}(U(\theta_{meta} + \sigma\mathbf{g}_i, T_i))$
    $v_i^- \leftarrow f^{T_i}(U(\theta_{meta} - \sigma\mathbf{g}_i, T_i))$
    $v_i \leftarrow \frac{1}{2}(v_i^+ - v_i^-)$
  end
  Update $\theta_{meta} \leftarrow \theta_{meta} + \frac{\beta}{\sigma n}\sum_{i=1}^{n} v_i\mathbf{g}_i$
end

IV. Evolutionary Meta-Learning in Noisy Environments

In Subsection IV-A, we first give a description of the noise model as well as an overview of our modifications to the HC operator. We then provide detailed analysis of these modifications and their comparisons in Subsection IV-B. In the following Section V, we describe how this method can be used for training adaptable policies for legged robots.

A. Overview

Our goal is to apply evolutionary meta-learning on legged robots in the real world. One bottleneck that we must address in real world settings is the abundance of noise, which may affect the trajectory of the Hill-Climbing adaptation. As a reasonable and general abstraction, we choose to model the noise ε *additively* in our evaluation of f as $\widetilde{f}(\theta, \varepsilon) = f(\theta) + \varepsilon$ and discuss its assumptions in Subsec. IV-B. In Table I, we generalize the original (termed "Sequential") Hill-Climbing method from [41]. We modify the 1-step update rule $\theta^{(q)} \to \theta^{(q+1)}$ to allow multiple $(P > 1)$ parallel evaluations. We derived two variants, *Average* and *Batch*, for dealing with noisy evaluations.

Sequential	$\theta^{(q+1)} = \underset{\theta\in\{\theta^{(q)}, \theta^{(q)}+\alpha\mathbf{g}\}}{\operatorname{argmax}} f(\theta)$
Average	$\theta^{(q+1)} = \underset{\theta\in\{\theta^{(q)}, \theta^{(q)}+\alpha\mathbf{g}\}}{\operatorname{argmax}} \frac{1}{P}\sum_{i=1}^{P} \widetilde{f}(\theta, \varepsilon_i)$
Batch	$\theta^{(q+1)} = \underset{\theta\in\{\theta^{(q)}, \theta^{(q)}+\alpha\mathbf{g}_1, \ldots, \theta^{(q)}+\alpha\mathbf{g}_P\}}{\operatorname{argmax}} \widetilde{f}(\theta, \varepsilon)$

TABLE I: We present the 3 Hill-Climbing steps for comparison: *Sequential, Average,* and *Batch*.

B. Adaptation Mechanisms

Average: A straightforward way to handle noise is to average noisy evaluations. We use empirical averages $\widehat{f}(\theta) = \frac{1}{P}\sum_{i=1}^{P}\widetilde{f}(\theta, \varepsilon_i)$ when evaluating a fixed parameter θ. However, this method has two limitations: (1) Low sample efficiency and (2) Restrictive noise assumptions.

Low sample efficiency. Real robot experiments are expensive. Assuming a fixed budget $(Q \cdot P)$ of *total* objective evaluations allowed during the adaptation, this averaging method sacrifices exploration for accuracy by lowering Q (the number of new parameters sampled) and increasing P (the number of parallel objective evaluations). Since it is often infeasible to measure the noise level in the real world, it is difficult to find a right balance between P and Q in practice.

Restrictive noise assumptions. This method inherently assumes the existence of a true *expected* objective $\mathbb{E}_{\varepsilon\sim\mathcal{D}}[\widetilde{f}(\theta, \varepsilon)]$ which can be approximated by $\widehat{f}(\theta)$, under a strict assumption that errors ε are i.i.d. sampled from a zero-mean distribution $\mathcal{D}$. The i.i.d. assumption can be unrealistic when experimenting in the real world where noise can easily be correlated (see Subsection VI-C for a detailed description in the Minitaur case). The zero-mean assumption is also unrealistic, as [2], which also touches on the Minitaur case, assumes the objective can be corrupted with *unbounded* magnitude *adversarially*, under bounds of the frequency of such corruptions. The adversarial assumption, which we will also use, avoids overly specific and complex modelling of the error by covering a variety of noise types using general *worst-case* theoretical guarantees. While [2] approaches this problem by estimating a biased gradient using regression, we must use an alternative method, as the Hill-Climbing method inherently does not estimate gradients. This leads to the following simple, yet effective adaptation step:

Batch: In order to maintain sample efficiency and reduce strong assumptions on noise, we propose the "Batch" Hill-Climbing method by modifying the adaptation operator to sample a *batch* of perturbations before performing the argmax, shown in Table I. Our batch method still maintains a diverse population of $(Q \cdot P)$ new parameters. Note that it takes the argmax based on *noisy* evaluations of the objective from P candidate parameters $\theta^{(q)} + \alpha\mathbf{g}_i,\ \forall i \in \{1, 2, ..., P\}$. This exact variant (termed *Gradientless Descent*) originates from [11], and was proven to possess convergence properties (for the convex/concave objective case) and is also independent to monotonic transformations of the reward (such as scaling) which can be useful in new test domains, when the model of the noise is generated by a bounded deterministic function $\varepsilon = h(\theta)$ of the input.

The noise in our real-robot setting exceeds even these assumptions, since the noise is non-deterministic and there can also be non-independent catastrophic failures. However, it turns out that Batch Hill-Climbing is robust even to this extreme level of noise. We assume that the reward estimates $f(\cdot)$ for all candidate permutations $\{\theta^{(q)} + \alpha\mathbf{g}_1, \ldots, \theta^{(q)} + \alpha\mathbf{g}_P\}$ may be corrupted with additive error $|\epsilon| \leq \Lambda$, and that up to W rewards may be corrupted with unbounded negative errors. If the number of candidates P is sufficiently large relative to W, then the average regret of the Batch Hill-Climbing algorithm can be bounded, up to irreducible error from Λ. That is, with high probability, we have $\frac{1}{Q}\sum_{q=0}^{Q}\left[f(\theta^{opt}) - f(\theta^{(q)})\right] \leq \frac{c_1}{\sqrt{Q}} + c_2$, where c_1, c_2 are constants depending on the concavity of the reward f, the size of the domain, and the level of noise Λ. We prove this novel claim formally in the Appendix of the arXiv version of this paper.

V. Application to Legged Robots

We use the Minitaur quadruped robot from Ghost Robotics [18] as our hardware platform.The Minitaur is actuated by 8 direct-drive motors, two for each leg. The robot is equipped with an IMU sensor to measure the orientation and the angular velocity of the robot's base, and motor encoders to measure its joint angles. In addition, we attach motion capture markers on the base of the robot. During the adaptation phase, we use a motion capture system to track the position of the robot to calculate the reward signal. We tethered the robot with a workstation, which runs the control loop at 166Hz. To validate our algorithm on the real robot, we first train a meta-policy in simulation, and then adapt the policy to two real-world tasks where the dynamics have changed significantly.

A. Training in Simulation

Since training in the real world is costly, we train our meta-policy in simulation and rely on our method to adapt the learned policy to various real world environments. The observation space includes the roll and pitch angle of the robot base, as well as all 8 motor angles, and a phase variable [22]. The action space consists of the desired swing and extension of each leg [43]. The reward function is

$$r(t) = \min(v, v_{\text{max}})dt - 0.005\sum_{i=1}^{8}\tau_i\omega_i dt$$

where v, v_{max} is the current and maximum forward velocity of the robot, τ_i, ω_i is the torque and angular velocity of each motor. The reward encourages the robot to walk at maximum speed, while penalizing energy expenditure. v_{max} starts at 0 when $t = 0$, increases linearly to 1.3m/s at 0.6s, and remains constant afterwards.

We leverage a physics simulation of the robot using PyBullet [4]. During training, each task samples a physics parameter from Table II. All tasks are trained with a maximum horizon of 500 steps (3 seconds). To prevent unsafe behaviors, we terminate an episode early if the center height

Parameter	Lower Bound	Upper Bound
Base Mass	75%	150%
Leg Mass	75%	150%
Battery Voltage	14.8V	16.8V
Motor Viscous Damping	0.0	0.02
Motor Strength	70%	100%
Contact Friction	0.75	1.5
Control Latency	0s	0.05s

TABLE II: Range of parameters randomized.

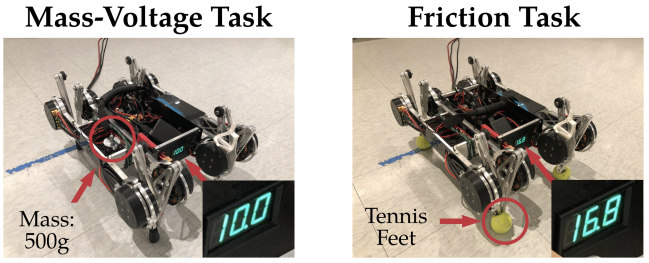


Fig. 2: Illustration of two real robot tasks, each with a red circle covering the physical modification, as well as depicting the battery voltages used.

drops below 13cm, or if the robot orientation deviates too far from the upright position. For ES-MAML, we use 300 workers for perturbations, and the same hyperparameters as found in [41]. For PG-MAML, we use the same implementation as [9] and tuned hyperparameters using gridsearch. For all experiments, our error bars are plotted with 1 standard deviation from the mean.

In order to pretrain our policy to effectively use a noise-reduction adaptation operator when applied to real-world tasks, we inject observation noise sampled from a unit Gaussian distribution in the simulated training environment, and also randomize the Minitaur's initialization.

B. Adaptation Tasks in the Real World

On the real robot, we test the adaptation of our method on two difficult tasks. Note that the changes between training and test environments are not only due to sim-to-real inaccuracies, but also include additional changes in battery level, center of mass, or foot friction.

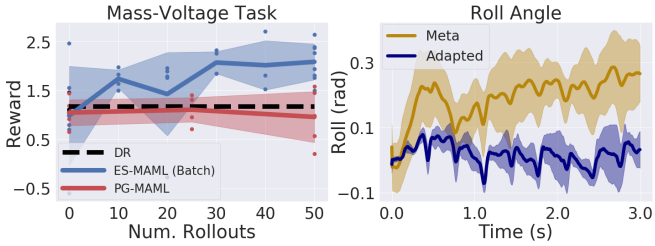
Mass-Voltage Task (Fig. 2, Left): In this task, we reduced the voltage of the power supply from 16.8V to 10V and added a mass of 500 grams on the right side of the robot (about 8% of robot mass). Note that although we randomized the battery voltage and base mass during training, the test voltage (10V) is far outside the training range (14.8V-16.8V) and the position of center-of-mass is not randomized.

Friction Task (Fig. 2, Right): To create a slippery contact, we replaced all rubber feet of the Minitaur robot with tennis balls, which significantly reduced the friction coefficient of contact.

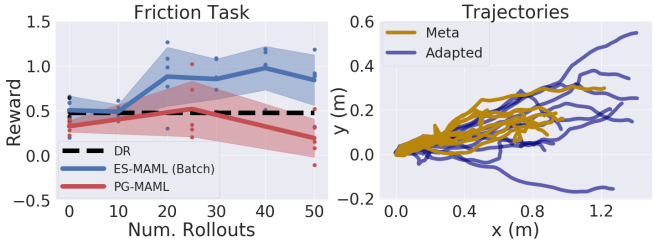
C. Real Robot Results

For the Mass-Voltage Task, the added weight tilts the robot to the right, which together with the reduced motor voltage, turns the robot towards right when we ran the meta-policy on the robot. Our method adapts the policy effectively within 50 real-world rollouts (150 seconds of data), improving the return by 100% (Fig. 3a, Left). The adaptation corrects the tilt, and returns the robot to a balanced roll angle (Fig. 3a, Right). As a result, the robot is able to walk farther and more straight as shown in Fig. 1 from Section I.

For the Friction Task, when applying the meta-policy, the extremely low friction causes the robot to slip constantly and leads to the chaotic trajectory distribution (Fig. 3, Left). The fine-tuned policy is still able to significantly increase forward walking distance, highlighting the superior performance of our method in noisy environments.



(a) Real robot result for **Mass-Voltage Task**. (**Left**): ES-MAML increases its performance as we use more rollouts during adaptation, outperforming domain randomization (DR) and PG-MAML. (**Right**): Roll Angle (deviation of Minitaur body from z-axis, to measure turning) is displayed for each of meta-policy and adapted policy from ES-MAML. Adaptation stabilizes the roll angle to ≈ 0, reducing turning.



(b) Real robot result for **Friction Task**. (**Left**): ES-MAML once again improves performance throughout Batch Hill-Climbing, while PG-MAML actually performs *worse* after 2 gradient steps of adaptation. (**Right**): Sample rollouts are displayed from both the meta-policy and adapted policy from ES-MAML, and shows that adaptation allows longer trajectories.

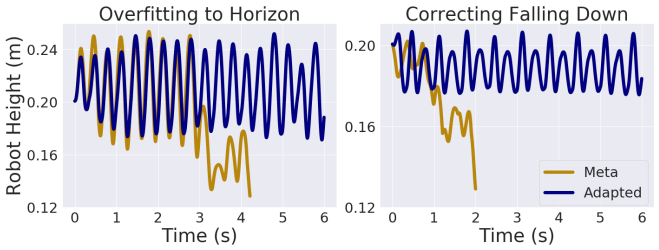
Fig. 3: Real robot results showing raw rewards and metrics collected from rollouts using our method (Batch Hill-Climbing) from both the meta-policy and adapted policy. For each task, we allow 50 rollouts for adaptation and set Batch Hill-Climbing to use $(Q, P) = (5, 10)$.

As a comparison, we trained the policy with Domain Randomization (DR) using ES. When using our method, although the meta policy achieves a similar performance as DR, the adapted policy significantly outperforms the DR policy. We also attempted adaptation with PG-MAML on the real robot using the same number of 50 rollouts (2 gradient step and 25 rollouts for each). For both tasks, although the meta policy performs similarly to that of ES-MAML, the policy does not improve after adaptation (Fig. 3a), and even performs worse in the Friction Task (Fig. 3b).

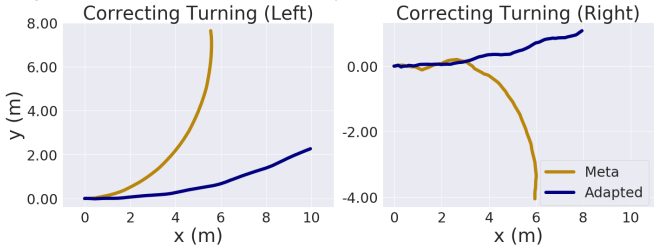
VI. Simulation Analysis and Discussion

To further demonstrate the effectiveness of our method, we perform additional ablation studies over several key components of our method in simulation. We aim to answer the following questions in our analysis:

- Does adaptation happen using our method? Qualitatively, how does the adaptation improve the performance?
- Can our method adapt more effectively and efficiently than PG-MAML?



(a) Adaptation corrects falling down, due to changes in horizon length, or unstable environment dynamics.



(b) Adaptation corrects turning, due to changes in environment variables such as battery voltages and motor strengths.

Fig. 4: Examples of ES-MAML adaptation in the Minitaur simulation environment. **(a)**: HC corrects the robot from falling, leading to longer and more stable walking behavior. Falling behavior can be caused by overfitting to horizon length (e.g. stopping halfway at time=3s), or unstable dynamics. **(b)**: HC corrects the robot from curving to the left or right, by straightening the walking trajectory.

- How does stochasticity in the environment affect training, and how does our Batch Hill-Climbing adaptation algorithm perform in noisy environments?

For a thorough evaluation, we performed a large amount of experiments in simulation to answer the above three questions.

A. Learned Adaptation Behaviors

To demonstrate that our algorithm can adapt policies to changes in dynamics, we use disjoint sets of random seeds for the training and testing environment populations, which generate environment parameters found in Table II. Furthermore, we increase the episode length of testing to 1000 steps from 500 used in training because we find that training can overfit to the episode length: the robot fails soon after 500 steps (Figure 4a, Left).

In new test environments, the meta-policy can fail in various ways depending on the environment parameters. For example, the robot falls before the episode ends, or veers off to the left or right (Figure 4b). Our method adapts effectively in the new environments; by using only 50 rollouts, by keeping the robot balanced while walking straight. Figure 5 shows that for a large variety of new environments, as the number of ES-MAML iterations increases, the adaption becomes more effective and achieves greater performance gains.

B. Comparison with PG-MAML

We perform comparisons between our method and PG-MAML across two different adaptation tasks, with more distribution difference between training and testing (Table III). In the *Uniform Test* setting, the training and testing environments sample different physical parameters within the same range. In the *Extreme Test* setting, the testing environment only samples physical parameters from the extremities of the range. As an extra baseline, we also train robust policies using Domain Randomization (DR) [43]. For a fair comparison, we use the same amount of experience (50 rollouts) for adaptation between ES-MAML and PG-MAML, where ES-MAML uses these for Hill-Climbing, while PG-MAML performs two gradient steps.

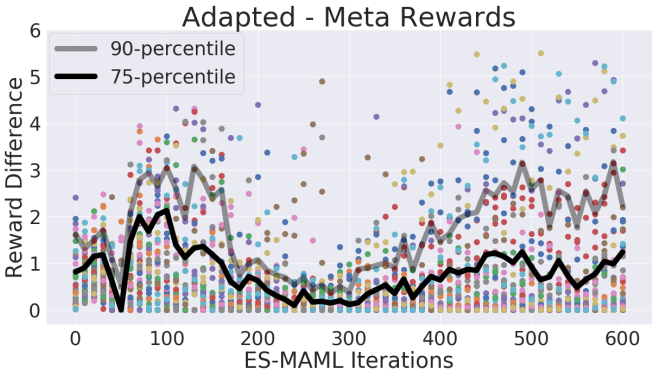


Fig. 5: Scatterplot showing adaptation gaps for 50 test tasks while training with ES-MAML. The same colored dots correspond to the same test task. Percentiles are plotted to display distribution of gaps.

Name	Train Task	Test Task
Uniform Test	Entire range	Entire range
Extreme Test	Entire range	$\{\min, \max\}$ of range

TABLE III: Test setup for PG-MAML Comparison. All environment parameters were sampled uniformly.

In both adaptation tasks, our method's adapted policy improves performance: The blue curve is always above the brown curve in Figure 6 and 7. As expected, the performance of the meta-policy decreases significantly in the Extreme Test (Fig. 7). Even on this more difficult environment, our method

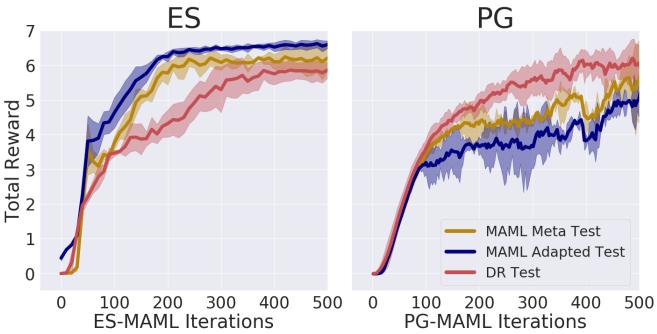


Fig. 6: **Uniform Test** for ES and PG. We see that ES-MAML's adaptation performance is the highest, outperforming its meta-policy and domain randomization with ES. On the other hand, the trend is reversed for PG-MAML, which possesses a negative adaptation gap and a very strong domain randomized policy.

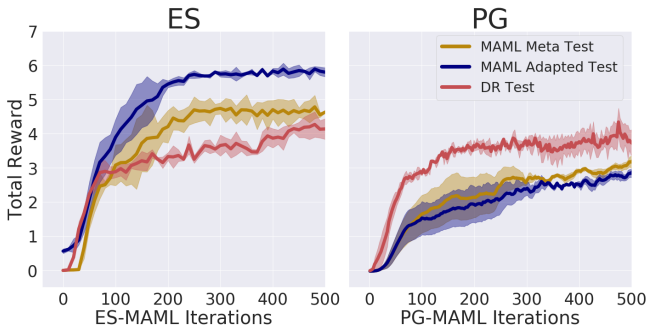


Fig. 7: **Extreme Test** for ES and PG. The extreme test environment increases the adaptation gap for ES-MAML, while PG-MAML produces lower gains everywhere, with its domain randomization performance remaining the strongest.

still adapts significantly and brings the adapted policy to a similar performance as Uniform Test.

In contrast for PG-MAML, the adaptation fails to perform any improvement in both cases. This is likely caused by noisy gradient estimation from policy gradient methods. This observation is consistent with poor adaptation performance on the real robot in Section V-B. While PG-MAML is well known for adapting to *reward* changes [48], its failure in this setting is potentially due to *dynamics* changes.

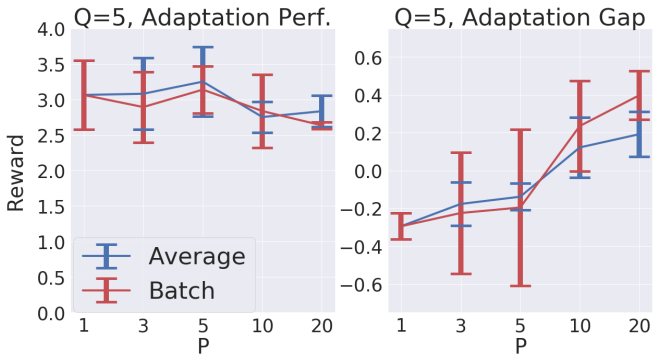
C. Adaptation in the Presence of Noise

We further compare our Hill-Climbing variants ("Average" and "Batch") in the stochastic Minitaur simulator (Fig. 8a). In the real robot experiments, we found that the sources of noise may include different initializations, drift of dynamics over time (e.g. due to motor overheating and battery level dropping), sensor noise, and environmental perturbations (e.g. ground with nonuniform textures and friction). To mimic real world scenarios, we further increase the stochasticity of our simulation. On top of observation noise and random initialization that are mentioned in Subsection V-A, we perturbed the robot with random forces so that the variance in rewards across multiple runs for the same policy is comparable to that observed on the real robot. In this highly noisy environment, we found that although the final performances are comparable, "Batch" Hill-Climbing generally produces more obvious adaptations than its "Average" counterpart. Increasing P generally improved adaptation, with "Batch" producing larger adaptation gains than "Average" when $P > 5$.

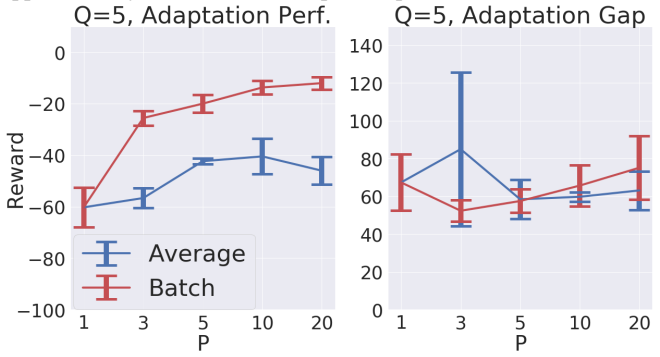
To demonstrate the generality of our method, we performed the same comparison in a different noisy environment. We created a noisy version of the widely-used Nav-2D environment from [9]. We added Gaussian noises (0.0 mean, 1.0 std.) to the observations while keeping all other setups unchanged. In Fig. 8b, "Batch" Hill-Climbing significantly improves final performance after adaptation, and the performance gain scales with P.

VII. Conclusion

Legged robots need to quickly adapt their locomotion skills to move in different dynamic environments. In this paper, we present an evolutionary meta-learning algorithm



(a) In **Minitaur-Noisy**, Batch produces a higher adaptation gap than Average, especially when increasing P, while still maintaining approximately same level of adaptation performance.



(b) In **Obs-Noise Nav-2D**, Batch outperforms Averaging when measuring mean adaptation performance across entire test distribution after training convergence.

Fig. 8: Comparison between Batch and Average Hill-Climbing on the Obs-Noise Nav-2D and Minitaur-Noisy environments. X-Axis: Parallel evaluations used during a Hill-Climbing adaptation step. Y-Axis: Total Reward.

that enables locomotion policies to quickly adapt in noisy real world scenarios. The core idea is to develop an efficient and noise-tolerant adaptation operator, and integrate it into meta-learning frameworks. We have shown that this Batch Hill-Climbing operator works better in handling noise than simply averaging rewards over multiple runs. Our algorithm has achieved greater adaptation performance than the state-of-the-art MAML algorithms that are based on policy gradient. Finally, we validate our method on a real quadruped robot. Trained in simulation, the locomotion policies can successfully adapt to two real-world robot environments, whose dynamics have been drastically changed.

In the future, we plan to extend our method in several ways. First, we believe that we can replace the Gaussian perturbations in the evolutionary algorithm with non-isotropic samples to further improve the sample efficiency during adaptation. With less robot data required for adaptation, we plan to develop a lifelong learning system, in which the robot can continuously collect data and quickly adjust its policy to learn new skills and to operate optimally in new environments.

ACKNOWLEDGEMENTS

We thank Vikas Sindhwani, Daniel Seita, and the Google AI Blog Team for fruitful discussions when writing this work.

REFERENCES

[1] Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. *arXiv:1909.12906*, 2019.

[2] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Deepali Jain, Yuxiang Yang, Atil Iscen, Jasmine Hsu, and Vikas Sindhwani. When random search is not enough: Sample-efficient and noise-robust blackbox optimization of RL policies. In *CoRL*, 2019.

[3] Paul F. Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. abs/1610.03518, 2016.

[4] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2020.

[5] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 2015.

[6] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. abs/1611.02779, 2016.

[7] Chrisantha Fernando, Jakub Sygnowski, Simon Osindero, Jane Wang, Tom Schaul, Denis Teplyashin, Pablo Sprechmann, Alexander Pritzel, and Andrei A. Rusu. Meta-learning by the baldwin effect. In *GECCO*, 2018.

[8] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*, 2018.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[10] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018.

[11] Daniel Golovin, John Karro, Greg Kochanski, Chansoo Lee, Xingyou Song, and Qiuyi Zhang. Gradientless descent: High-dimensional zeroth-order optimization. In *ICLR*, 2020.

[12] Abhishek Gupta, Russell Mendonca, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *NeurIPS*, 2018.

[13] Sehoon Ha, Joohyung Kim, and Katsu Yamane. Automated deep reinforcement learning environment for hardware of a modular legged robot. In *International Conference on Ubiquitous Robots (UR)*, 2018.

[14] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. In *RSS*, 2019.

[15] Josiah P Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *AAAI*, 2017.

[16] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly C. Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *NeurIPS*, 2018.

[17] 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*, 2019.

[18] Gavin Kenneally, Avik De, and Daniel E Koditschek. Design principles for a family of direct-drive legged robots. *IEEE RA-L*, 2016.

[19] Nate Kohl and Peter Stone. Machine learning for fast quadrupedal locomotion. In *AAAI*, 2004.

[20] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *ICRA*, 2004.

[21] Tianyu Li, Hartmut Geyer, Christopher G Atkeson, and Akshara Rai. Using deep reinforcement learning to learn high-level policies on the atrias biped. In *ICRA*, 2019.

[22] Tianyu Li, Nathan Lambert, Roberto Calandra, Franziska Meier, and Akshara Rai. Learning generalizable locomotion skills with hierarchical reinforcement learning. In *ICRA*, 2020.

[23] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[24] Hao Liu, Richard Socher, and Caiming Xiong. Taming MAML: efficient unbiased meta-reinforcement learning. In *ICML*, 2019.

[25] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. In *NeurIPS*, 2018.

[26] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. Active domain randomization. *arXiv:1904.04762*, 2019.

[27] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[28] Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang Gu, and Vikash Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv:1908.05224*, 2019.

[29] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019.

[30] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. abs/1803.02999, 2018.

[31] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, 2018.

[32] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. Towards generalization and simplicity in continuous control. In *NeurIPS*, 2017.

[33] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, 2019.

[34] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. In *ICLR*, 2019.

[35] Fereshteh Sadeghi and Sergey Levine. CAD2RL: real single-image flight without a single real image. In *RSS*, 2017.

[36] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. abs/1703.03864, 2017.

[37] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. abs/1707.06347, 2017.

[39] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[40] Abhik Singla, Shounak Bhattacharya, Dhaivat Dholakiya, Shalabh Bhatnagar, Ashitava Ghosal, Bharadwaj Amrutur, and Shishir Kolathaya. Realizing learned quadruped locomotion behaviors through kinematic motion primitives. In *ICRA*, 2019.

[41] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. In *ICLR*, 2020.

[42] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 1999.

[43] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *RSS*, 2018.

[44] Russ Tedrake, Teresa Weirui Zhang, and H Sebastian Seung. Learning to walk in 20 minutes.

[45] Emo Todorov. Mujoco. http://mujoco.org, 2015–2020.

[46] Jane X. Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z. Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt M. Botvinick. Learning to reinforcement learn. In *CogSci*, 2017.

[47] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. 2019.

[48] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Jie Tan, and Chelsea Finn. Norml: No-reward meta learning. In *AAMAS*, 2019.

[49] Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*, 2017.

[50] Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. 2019.

[51] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. In *ICRA*, 2020.

[52] Luisa M. Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep RL via meta-learning. abs/1910.08348, 2019.