Learning to Adapt in Dynamic, Real-World Environments via Meta-Reinforcement Learning

Anusha Nagabandi *
UC Berkeley

Ignasi Clavera*
UC Berkeley

Simin Liu
UC Berkeley

Ronald S. Fearing
UC Berkeley

Pieter Abbeel
UC Berkeley

Sergey Levine
UC Berkeley

Chelsea Finn
UC Berkeley

1 Introduction

Both model-based and model-free reinforcement learning (RL) methods generally operate in one of two regimes: all training is performed in advance, producing a model or policy that can be used at test-time to make decisions in settings that approximately match those seen during training; or, training is slowly performed online. However, in both of these cases, dynamic changes such as failure of a robot's components, encountering a new terrain, or other unexpected perturbations, can cause the agent to fail. In contrast, humans can rapidly adapt their behavior to unseen physical perturbations and changes in their dynamics [1]: If an agent has encountered a large number of perturbations in the past, it can in principle use that experience to *learn how to adapt*. In this work, we propose a meta-learning approach for learning online adaptation.

Although RL methods can achieve impressive results in simulation [8, 6, 9], the real world presents two major challenges: sampling is exceedingly expensive, and unseen situations cause proficient but specialized policies to fail at test time. Given that it is impractical to train separate policies to accommodate all possible situations, this work proposes to learn how to quickly and effectively adapt online to new tasks. We develop a model-based meta-RL algorithm, which allows for much greater sample efficiency [2] than model-free meta-RL approaches [3, 11, 4]. Our approach foregoes the episodic framework on which model-free meta-RL approaches rely on. Instead, our method considers each timestep to potentially be a new "task;" this allows a task to represent anything from different parts of the state space, to experiencing disturbances, or attempting a new goal. As a result, our approach alleviates a central challenge of model-based reinforcement learning: acquiring an accurate global model.

The primary contribution of our work is an efficient method for online adaptation, and to the best of our knowledge, it is the first meta-RL algorithm to be applied on a real robotic system. Our algorithm efficiently trains a global model that is capable of using its recent experiences to quickly adapt, achieving fast online adaptation in dynamic environments. We evaluate two versions of our approach, recurrence-based adaptive learner (ReBAL) and gradient-based adaptive learner (GrBAL) on stochastic continuous control tasks with complex contact dynamics (Fig. 2). Our method attains substantial improvement over prior approaches, and our experiments demonstrate online adaptation on simulated agents (adapting online to novel terrains, crippled body parts, and highly-dynamic environments) as well as a real dynamic legged millirobot (adapting online to a missing leg, novel terrains and slopes, errors in pose estimation, and pulling payloads).

Figure 1: We implement our sample-efficient meta-reinforcement learning algorithm on a real legged millirobot, enabling online adaptation to new tasks and unexpected occurrences such as losing a leg.

*Equal contribution

2nd Workshop on Meta-Learning at NeurIPS 2018, Montréal, Canada.

2 Meta-Learning for Online Model Adaptation

In this section, we present our approach for meta-learning for online model adaptation. Standard meta-learning formulations require the learned model $p_{\boldsymbol{\theta}_*}$ to adapt, using an update rule $u_{\boldsymbol{\psi}_*}$ after seeing M data points from some new "task." Our notion of task is slightly more fluid, where every segment of a trajectory can be considered to be a different "task," and the past M observations can be considered as providing more information about the current task setting. Since changes in system dynamics, terrain details, or other environmental changes can occur at any time, we consider (at every time step) the problem of adapting to the M past time steps.

We use the notion of environment $\mathcal{E}$ to denote different settings or configurations of a particular problem. We assume a distribution of environments $\rho(\mathcal{E})$ that share some common structure, such as the same observation and action space, but may differ in their dynamics $p_{\mathcal{E}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. A sequence of states and actions is denoted by $\tau_{\mathcal{E}}(i, j) = (\mathbf{s}_i, \mathbf{a}_i, ..., \mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1})$. We pose the meta-RL problem in this setting as an optimization over $(\boldsymbol{\theta}, \boldsymbol{\psi})$ with respect to a maximum likelihood meta-objective. The meta-objective is the likelihood of the data under a predictive model $\hat{p}_{\boldsymbol{\theta}'}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ with parameters $\boldsymbol{\theta}'$, where $\boldsymbol{\theta}' = u_{\boldsymbol{\psi}}(\tau_{\mathcal{E}}(t-M, t-1), \boldsymbol{\theta})$ corresponds to model parameters that were updated using the past M data points. Concretely, this corresponds to the following optimization:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\psi}} \mathbb{E}_{\substack{\mathcal{E} \sim \rho(\mathcal{E}) \\ t \sim \mathcal{U}([T-K-1])}} \Big[\mathcal{L}(\tau_{\mathcal{E}}(t, t+K), \boldsymbol{\theta}'_{\mathcal{E}})\Big] \quad \text{s.t.:} \quad \boldsymbol{\theta}'_{\mathcal{E}} = u_{\boldsymbol{\psi}}(\tau_{\mathcal{E}}(t-M, t-1), \boldsymbol{\theta}), \tag{1}$$

where the loss $\mathcal{L}$ corresponds to the negative log likelihood of the data under the model:

$$\mathcal{L}(\tau_{\mathcal{E}}(t, t+K), \boldsymbol{\theta}'_{\mathcal{E}}) \triangleq -\frac{1}{K} \sum_{k=t}^{t+K} \log \hat{p}_{\boldsymbol{\theta}'_{\mathcal{E}}}(\mathbf{s}_{k+1}|\mathbf{s}_k, \mathbf{a}_k). \tag{2}$$

In the meta-objective in Equation 1, note that the past M points are used to adapt $\boldsymbol{\theta}$ into $\boldsymbol{\theta}'$, and the loss of this $\boldsymbol{\theta}'$ is evaluated on the future K points. Thus, we use the past M timesteps to provide insight into how to adapt our model to perform well for nearby future timesteps. As outlined in Algorithm 1, the update rule $u_{\boldsymbol{\psi}}$ for the inner update and a gradient step on $\boldsymbol{\theta}$ for the outer update allow us to optimize this meta-objective of adaptation. Thus, we achieve fast adaptation at test time by being able to fine-tune the model using just M data points.

While we focus on reinforcement learning problems in our experiments, this meta-learning approach could be used for a learning to adapt online in a variety of sequence modeling domains. We present our algorithm using both a recurrence and a gradient-based meta-learner, as we discuss next.

Gradient-Based Adaptive Learner (GrBAL). GrBAL uses a gradient-based meta-learning to perform online adaptation; in particular, we use MAML [4]. In this case, our update rule is prescribed by gradient descent (Eq. 3)

$$\boldsymbol{\theta}'_{\mathcal{E}} = u_{\boldsymbol{\psi}}(\tau_{\mathcal{E}}(t-M, t-1), \boldsymbol{\theta}) = \boldsymbol{\theta}_{\mathcal{E}} + \boldsymbol{\psi} \nabla_{\boldsymbol{\theta}} \frac{1}{M} \sum_{m=t-M}^{t-1} \log \hat{p}_{\boldsymbol{\theta}_{\mathcal{E}}}(\mathbf{s}_{m+1}|\mathbf{s}_m, \mathbf{a}_m) \tag{3}$$

Recurrence-Based Adaptive Learner (ReBAL). ReBAL, instead, utilizes a recurrent model, which learns its own update rule (i.e., through its internal gating structure. In this case, $\boldsymbol{\psi}$ and $u_{\boldsymbol{\psi}}$ corresponds to the weights of the recurrent model that update its hidden state.

3 Model-Based Meta-Reinforcement Learning

Now that we have discussed our approach for enabling online adaptation, we next propose how to build upon this idea to develop a model-based meta-reinforcement learning algorithm. Given $\boldsymbol{\theta}_*$ and $\boldsymbol{\psi}_*$, the agent uses its recent experience to adapt the model parameters: $\boldsymbol{\theta}'_* = u_{\boldsymbol{\psi}_*}(\tau(t-M, t), \boldsymbol{\theta}_*)$. This results in a model $\hat{p}_{\boldsymbol{\theta}'_*}$ that better captures the local dynamics in the current setting, task, or environment. This adapted model is then passed to our controller, along with the reward function r and a planning horizon H. We use model predictive path integral control (MPPI) [12], but, in principle, our model adaptation approach is agnostic to the model predictive control (MPC) method used. This use of MPC can help compensate for model inaccuracies by preventing accumulating errors, since we replan at each time step using both updated state information and an updated model $\hat{p}_{\boldsymbol{\theta}'_{\mathcal{E}}}$. See Algorithm 2 for this full adaptation procedure, and Algorithm 1 for the full meta-training procedure, which uses Algorithm 2 to produce the on-policy rollouts used for model-bootstrapping.

Algorithm 1 Model-Based Meta-RL (train)

Require: Distribution $\rho_{\mathcal{E}}$ over tasks
Require: Learning rate $\beta \in \mathbb{R}^+$
Require: Number of sampled tasks N, dataset $\mathcal{D}$
Require: Task sampling freq. $n_S \in \mathbb{Z}^+$

Randomly initialize $\boldsymbol{\theta}$
for $i = 1, \ldots$ **do**
  if $i \bmod n_S = 0$ **then**
    Sample $\mathcal{E} \sim \rho(\mathcal{E})$
    Collect $\tau_{\mathcal{E}}$ using Alg. 2
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_{\mathcal{E}}\}$
  end if
  for $j = 1 \ldots N$ **do**
    $\tau_{\mathcal{E}}(t-M, t-1), \tau_{\mathcal{E}}(t, t+K) \sim \mathcal{D}$
    $\boldsymbol{\theta}'_{\mathcal{E}} \leftarrow u_{\boldsymbol{\psi}}(\tau_{\mathcal{E}}(t-M, t-1), \boldsymbol{\theta})$
    $\mathcal{L}_j \leftarrow \mathcal{L}(\tau_{\mathcal{E}}(t, t+K), \boldsymbol{\theta}'_{\mathcal{E}})$
  end for
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{j=1}^{N} \mathcal{L}_j$
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \frac{1}{N} \sum_{j=1}^{N} \mathcal{L}_j$
end for
Return $(\boldsymbol{\theta}, \boldsymbol{\psi})$ as $(\boldsymbol{\theta}_*, \boldsymbol{\psi}_*)$

Algorithm 2 Online Model Adaptation (test)

Require: Meta-learned parameters $\boldsymbol{\theta}_*, \boldsymbol{\psi}_*$
Require: controller(), H, r, n_A

$\mathcal{D} \leftarrow \emptyset$
for each timestep t **do**
  $\boldsymbol{\theta}'_* \leftarrow u_{\boldsymbol{\psi}_*}(\mathcal{D}(t-M, t-1), \boldsymbol{\theta}_*)$
  $\mathbf{a} \leftarrow$ controller$(\boldsymbol{\theta}'_*, r, H, n_A)$
  Execute $\mathbf{a}$, add result to $\mathcal{D}$
end for
Return rollout $\mathcal{D}$

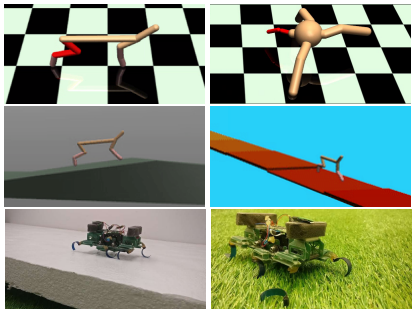

Figure 2: Two real-world and four simulated environments on which our method is evaluated and adaptation is crucial for success (e.g., adapting to different slopes and leg failures)

4 Results

We aim to answer the following questions: (1) how does our approach compare to meta model-free RL in sample efficiency and performance, (2) does our approach enable fast adaptation, both inside and outside of the training distribution, and (3) can our method learn to adapt online on a real robot? For our experiments, we use simulated half-cheetah and ant agents (Fig. 2), using the MuJoCo physics engine [10]. We specify training and testing details in the appendix.

4.1 Sample Efficiency

We first evaluate our model-based meta-RL methods (GrBAL and ReBAL) against model-free RL, model-free meta-RL, and model-based RL methods. See appendix for details on these methods. Figure 3 shows average return across test environments w.r.t. the amount of data used for meta-training. We meta-train the model-free methods (TRPO and MAML-RL) until convergence, using the equivalent of about two days of real-world experience. In contrast, we meta-train the model-based methods (including ours) using the equivalent of 1.5-3 hours of data. Our methods result in superior or equivalent performance to the model-free agent that is trained with 1000 times more data. Our methods also surpass the performance of the non-meta-learned model-based approaches. Finally, our performance closely matches the high asymptotic performance of the model-free meta-RL method for half-cheetah disabled, and achieves a suboptimal performance for ant crippled (but does so with 1000 times less data).

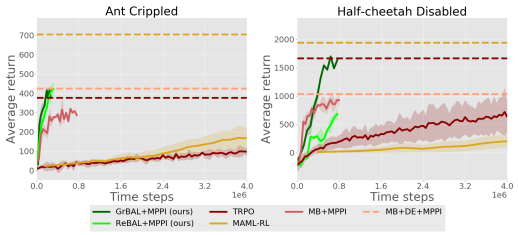

Figure 3: Compared to other methods, our model-based meta-RL methods achieve good performance with 1000 $\times$ less data. Dotted lines indicate performance at convergence.

4.2 Fast Adaptation & Generalization

We further compare various methods from above in the low-data regime (1.5-3 hours of real-world experience). We compare GrBAL and ReBAL against the following three methods: a non-adaptive MB method ("MB"), an adaptive model-based method ("MB + DE") that uses dynamic-evaluation at run time, and a model-free method ("TRPO"). We also provide the performance of a MB oracle,

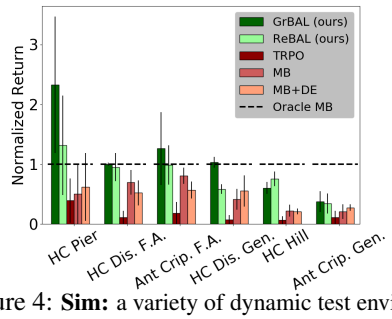

Figure 4: **Sim:** a variety of dynamic test environments, showing the difficulty of training a global model and the importance of adaptation. Results are normalized so MB oracle achieves 1.

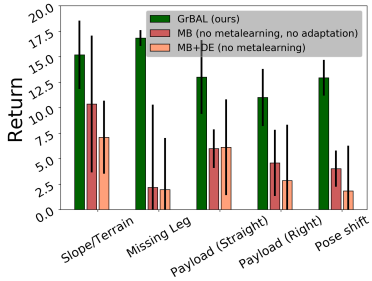

Figure 5: **Real-world:** GrBAL outperforms both MB and MB+DE when tested on tasks that require online adaptation and/or were never seen during training.

trained using unlimited data from the single test task (and thus requiring no generalization). We test the ability of each approach to adapt to sudden changes in the environment (F.A. = fast adaptation), as well as to generalize (Gen.) beyond the training environments.

As shown in Fig. 4, TRPO performs poorly in the low data regime. Although MB+DE achieves better generalization than MB, the slow nature of its adaptation causes it to fall behind MB in the environments that require fast adaptation. Our approach surpasses the other approaches in all tasks; in fact, it surpasses even the model-based oracle in the HC pier and the F.A. ant environments, showing the need for adaptation in stochastic environments, where even a model trained with a lot of data cannot be robust to unexpected occurrences or disturbances. ReBAL displays strengths on tasks where longer sequential inputs allow it to better asses current task settings, but overall, GrBAL seems to perform better.

4.3 Real-World Results

To test our meta-MB RL method's sample efficiency, as well as its ability to perform fast and effective online adaptation, we applied GrBAL to a real legged millirobot. This small 6-legged robot, as shown in Fig. 1 and Fig. 2, presents a modeling and control challenge in the form of highly stochastic and dynamic movement, and is an excellent candidate for online adaptation for many reasons (see the appendix for more details on the system itself). As above, we compare GrBAL to a model-based method (MB) that involves neither meta-training nor online adaptation, as well as a dynamic evaluation method that involves online adaptation of that MB model (MB+DE).

We meta-train a dynamics model for this robot (entirely on real-world data) using the meta-objective described in Equation 1. We see that GrBAL has comparable performance on terrains from the training distribution (see appendix), but greatly outperforms (Fig. 5) the other methods when faced with unseen and/or changing tasks at test time. Fig. 6 shows that unlike MB and MB+DE, GrBAL can quickly 1) adapt online to a missing leg, 2) adjust to novel terrains and slopes, 3) account for miscalibration or errors in pose estimation, and 4) compensate for pulling payloads.

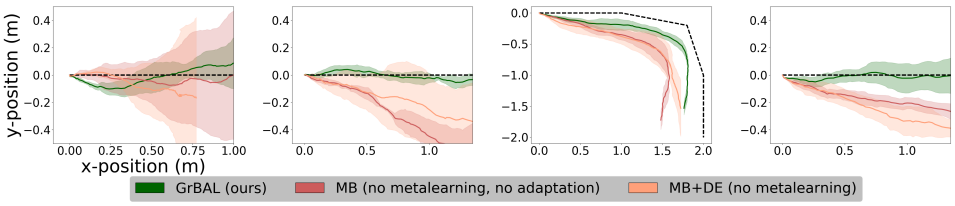

Figure 6: By effectively adapting online, our method performs well on new unseen tasks that require fast and effective online adaptation for success. The dotted black line indicates the desired trajectory in the xy plane.

5 Conclusion

In this work, we present an approach for model-based meta reinforcement learning that enables fast, online adaptation in dynamic environments. We show that meta-learning a model for online adaptation results in a method that is able to adapt to unseen situations or sudden and drastic changes in the environment, and is also sample efficient to train. We provide two instantiations of our approach (ReBAL and GrBAL), and we provide a comparison with other prior methods on a range of continuous control tasks. We show that our approach is practical for real-world in contrast to less efficient model-free meta-reinforcement learning approaches, and that the capability to adapt quickly is particularly important under complex real-world dynamics.

References

[1] D. A. Braun, A. Aertsen, D. M. Wolpert, and C. Mehring. Learning optimal adaptation strategies in unpredictable motor tasks. *Journal of Neuroscience*, 2009.

[2] M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[3] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl$ˆ2$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.

[4] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[5] B. Krause, E. Kahembwe, I. Murray, and S. Renals. Dynamic evaluation of neural sequence models. *CoRR*, abs/1709.07432, 2017.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[7] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *CoRR*, abs/1708.02596, 2017.

[8] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015.

[9] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 2017.

[10] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

[11] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[12] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *CoRR*, abs/1509.01149, 2015.

A Comparisons

In our experiments, we compare our model-based meta-RL methods (GrBAL and ReBAL) to several prior methods:

Model-free RL (TRPO): To evaluate the importance of adaptation, we compare to a model-free RL agent that is trained across environments $\mathcal{E} \sim \rho(\mathcal{E})$ using TRPO [8].

Model-free meta-RL (MAML-RL): We compare to a state-of-the-art model-free meta-RL method, MAML-RL [4].

Model-based RL (MB): Similar to the model-free agent, we also compare to a single model-based RL agent, to evaluate the importance of adaptation. This model is trained using supervised model-error and iterative model bootstrapping.

Model-based RL with dynamic evaluation (MB+DE): We compare to an agent trained with model-based RL, as above. However, at test time, the model is adapted by taking a gradient step at each timestep using the past M observations, akin to dynamic evaluation [5]. This final comparison evaluates the benefit of explicitly training for adaptability.

All model-based approaches (MB, MB+DE, GrBAL, and ReBAL) use model bootstrapping (i.e., iterations of aggregating data from on-policy rollouts and using that data to re-train the model). They also use the same neural network architecture and the same planner within experiments: MPPI [12] for the simulated experiments and random shooting (RS) [7] for the real-world experiments.

B Environments

We conduct experiments on a variety of simulated robots using the MuJoCo physics engine. For all of our environments, we model the transition probabilities as Gaussian random variables with mean parameterized by the neural network model, and fixed variance. In this case, maximum likelihood estimation corresponds to minimizing the mean squared error. Here, we present the details of the simulated tasks from the result sections.

Half-cheetah (HC): disabled joint. For each meta-training rollout, we randomly sample a joint to be disabled (i.e., the agent cannot apply torques to that joint). At test time, we evaluate performance in three different situations: (a) disabling a joint seen at train time, (b) disabling an unseen joint, and (c) switching between disabled joints during a rollout.

HC: sloped terrain. During meta-training, we choose terrain of varying gentle upward and downward slopes. In this task, it is especially important to incorporate past experience into the model, since the cheetah has no means of directly observing the incline. At test time, we evaluate performance on (d) a gentle upward slope, (e) a steep hill that goes up and down, and (f) a steep upward slope.

HC: pier. In this task, the cheetah runs over a series of blocks that are floating on water. Each block moves up and down when stepped on, and the changes in the dynamics are drastic and rapid, due to each block having different damping and friction properties. The HC is meta-trained on varying these block properties, and tested on (g) a specific configuration of block properties.

Ant: crippled leg. For each meta-training rollout, we randomly sample a leg on a quadrupedal robot and disable it. Disabling a leg unexpected drastically changes the dynamics. We evaluate on (h) crippling a leg from the training distribution, (i) crippling a leg from outside the training distribution, and (j) crippling a leg in the middle of a rollout.

7-DoF arm: force perturbations. We train a 7-DoF robot arm to carry an object to a goal position while applying random perturbation forces to the object. At test time, we evaluate with (k) a constant low force to the object, (l) a force $3\times$ stronger than during training, and (m) a force that randomly changes every 50 time-steps. This allow us to evaluate the ability of our method to adapt online to perturbations that clearly lie outside the training distribution.

C Real robot

To test our meta-MB RL method's sample efficiency, as well as its ability to perform fast and effective online adaptation, we applied GrBAL to a real legged millirobot. Due to the cost of running real robot

experiments, we choose the better performing method (i.e., GrBAL) to evaluate on the real robot. This small 6-legged robot presents a modeling and control challenge in the form of highly stochastic and dynamic movement. This robot is an excellent candidate for online adaptation for many reasons: the rapid manufacturing techniques and numerous custom-design steps used to construct this robot make it impossible to reproduce the same dynamics each time, its linkages and other body parts deteriorate over time, and it moves very quickly and dynamically with bounding-style gaits; hence, its dynamics are strongly dependent on the terrain or task at hand.

The state space of the robot is a 24-dimensional vector, including center of mass positions and velocities, center of mass pose and angular velocities, back-EMF readings of motors, encoder readings of leg motor angles and velocities, and battery voltage. We define the action space to be velocity setpoints of the rotating legs. The action space has a dimension of two, since one motor on each side is coupled to all three of the legs on that side. All experiments are conducted in a motion capture room. Computation is done on an external computer, and the velocity setpoints are streamed over radio at 10 Hz to be executed by a PID controller on the microcontroller on-board of the robot.

We collect approximately 30 minutes of data from each of the three training terrains (carpet, styrofoam, turf). This data was entirely collected using a random policy, in conjunction with a safety policy, whose sole purpose was to prevent the robot from exiting the area of interest.

Table 1 shows that after training the agent on 30 minutes of random data from three different terrains, GrBAL achieves comparable trajectory following costs to the baselines (on the terrains seen during training). Note that results of GrBAL on tasks that are further outside of the training distribution are presented and discussed in the results section of the paper.

		Left	Str	Z-z	F-8
Carpet	GrBAL	4.07	3.26	7.08	5.28
	MB	3.94	3.26	6.56	5.21
Styrofoam	GrBAL	3.90	3.75	7.55	6.01
	MB	4.09	4.06	7.48	6.54
Turf	GrBAL	1.99	1.65	2.79	3.40
	MB	1.87	1.69	3.52	2.61

Table 1: Trajectory following costs for real-world GrBAL and MB results, showing comparable performance on terrains from the training distribution.

D Effect of Meta-Training Distribution

To see how training distribution affects test performance, we ran an experiment that used GrBAL to train models of the 7-DOF arm, where each model was trained on the same number of datapoints during meta-training, but those datapoints came from different ranges of force perturbations. We observe (in the plot below) that

1. Seeing more during training is helpful during testing — a model that saw a large range of force perturbations during training performed the best

2. A model that saw no perturbation forces during training did the worst

3. The middle 3 models show comparable performance in the "constant force = 4" case, which is an out-of-distribution task for those models. Thus, there is not actually a strong restriction on what needs to be seen during training in order for adaptation to occur at train time (though there is a general trend that more is better)

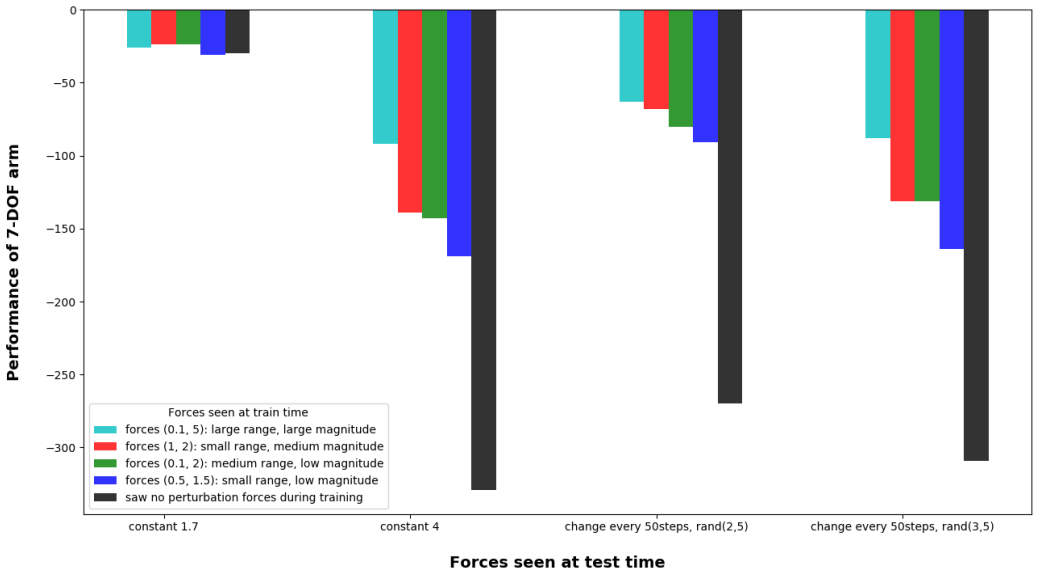

Figure 7: Effect of the meta-training distribution on test performance

E Model Prediction Errors: Pre-update vs. Post-update

In this section, we show the effect of adaptation in the case of GrBAL. In particular, we show the histogram of the K step normalized error, as well as the per-timestep visualization of this error during a trajectory. Across all tasks and environments, the post-updated model $\hat{p}_{\boldsymbol{\theta}'_*}$ achieves lower prediction error than the pre-updted model $\hat{p}_{\boldsymbol{\theta}_*}$.

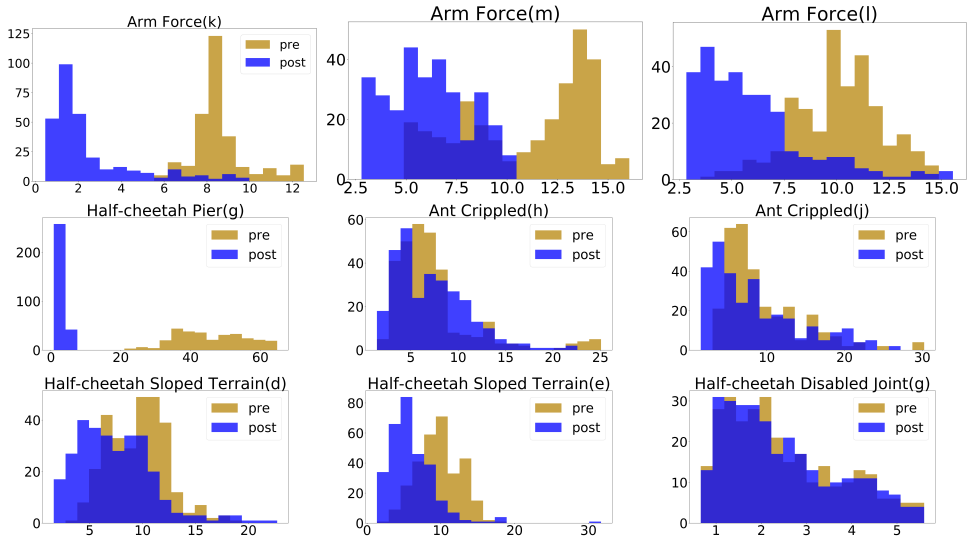

Figure 8: Histogram of the K step normalized error across different tasks. GrBAL accomplishes lower model error when using the parameters given by the update rule.

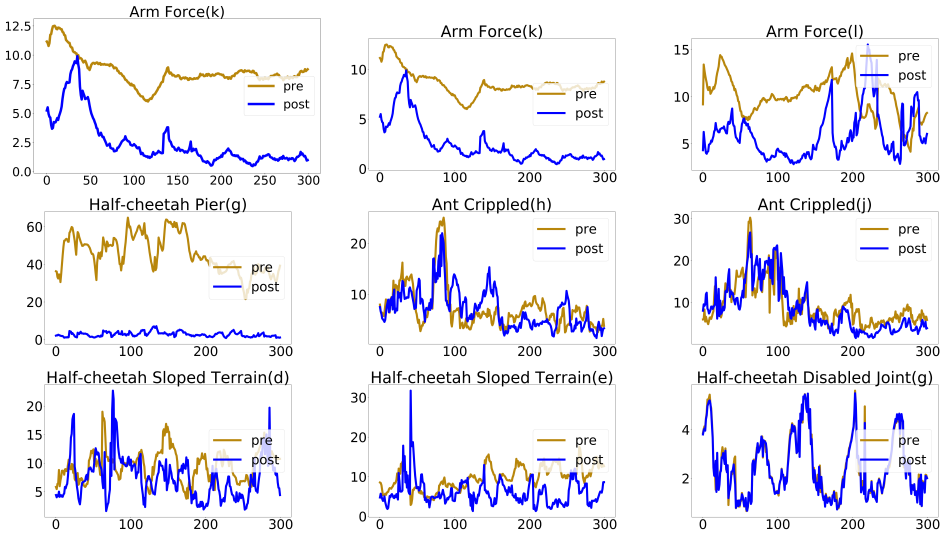

Figure 9: At each time-step we show the K step normalized error across different tasks. GrBAL accomplishes lower model error using the parameters given by the update rule.

F Reward functions

For each MuJoCo agent, the same reward function is used across its various tasks. Table 2 shows the reward functions used for each agent. We denote by x_t the x-coordinate of the agent at time t, $\boldsymbol{ee}_t$ refers to the position of the end-effector of the 7-DoF arm, and $\boldsymbol{g}$ corresponds to the position of the desired goal.

Table 2: Reward functions

	Reward function
Half-cheetah	$\frac{x_{t+1}-x_t}{0.01} - 0.05\|\boldsymbol{a}_t\|_2^2$
Ant	$\frac{x_{t+1}-x_t}{0.0e} - 0.005\|\boldsymbol{a}_t\|_2^2 + 0.05$
7-DoF Arm	$-\|\boldsymbol{ee}_t - \boldsymbol{g}\|_2^2$

G Hyperparameters

Below, we list the hyperparameters of our experiments. In all experiments we used a single gradient step for the update rule of GrBAL. The learning rate (LR) of TRPO corresponds to the Kullback–Leibler divergence constraint. # Task/itr corresponds to the number of tasks sampled for collecting data to train the model or model, whereas # TS/itr is the total number of times steps collected (for all tasks). Finally, T refers to the horizon of the task.

Table 3: Hyperparameters for the half-cheetah tasks

	LR	Inner LR	Epochs	K	M	Batch Size	# Tasks/itr	# TS/itr	T	n_A Train	H Train	n_A Test	H Test
GrBAL	0.001	0.01	50	32	32	500	32	64000	1000	1000	10	2500	15
ReBAL	0.001	-	50	32	32	500	32	64000	1000	1000	10	2500	15
MB	0.001	-	50	-	-	500	64	64000	1000	1000	10	2500	15
TRPO	0.05	-	-	-	-	50000	50	50000	1000	-	-	-	-

Table 4: Hyperparameters for the ant tasks

	LR	Inner LR	Epochs	K	M	Batch Size	# Tasks/itr	# TS/itr	T	n_A Train	H Train	n_A Test	H Test
GrBAL	0.001	0.001	50	10	16	500	32	24000	500	1000	15	1000	15
ReBAL	0.001	-	50	32	16	500	32	32000	500	1000	15	1000	15
MB	0.001	-	70	-	-	500	10	10000	500	1000	15	1000	15
TRPO	0.05	-	-	-	-	50000	50	50000	500	-	-	-	-

Table 5: Hyperparameters for the 7-DoF arm tasks

	LR	Inner LR	Epochs	K	M	Batch Size	# Tasks/itr	# TS/itr	T	n_a Train	H Train	n_a Test	H Test
GrBAL	0.001	0.001	50	32	16	1500	32	24000	500	1000	15	1000	15
ReBAL	0.001	-	50	32	16	1500	32	24000	500	1000	15	1000	15
MB	0.001	-	70	-	-	10000	10	10000	500	1000	15	1000	15
TRPO	0.05	-	-	-	-	50000	50	50000	500	-	-	-	-