Fast Online Adaptation in Robotics through Meta-Learning Embeddings of Simulated Priors

Rituraj Kaushik, Timothée Anne and Jean-Baptiste Mouret*

Abstract—Meta-learning algorithms can accelerate the model-based reinforcement learning (MBRL) algorithms by finding an initial set of parameters for the dynamical model such that the model can be trained to match the actual dynamics of the system with only a few data-points. However, in the real world, a robot might encounter any situation starting from motor failures to finding itself in a rocky terrain where the dynamics of the robot can be significantly different from one another. In this paper, first, we show that when meta-training situations (the prior situations) have such diverse dynamics, using a single set of meta-trained parameters as a starting point still requires a large number of observations from the real system to learn a useful model of the dynamics. Second, we propose an algorithm called FAMLE that mitigates this limitation by meta-training several initial starting points (i.e., initial parameters) for training the model and allows the robot to select the most suitable starting point to adapt the model to the current situation with only a few gradient steps. We compare FAMLE to MBRL, MBRL with a meta-trained model with MAML, and model-free policy search algorithm PPO for various simulated and real robotic tasks, and show that FAMLE allows the robots to adapt to novel damages in significantly fewer time-steps than the baselines.

I. INTRODUCTION

Reinforcement learning (RL) algorithms have shown many promising results, starting from playing Atari games from observing pixels or defeating professional Go players. However, these impressive successes were possible due to enormous interaction time with the games in simulation. For instance, in [1], around 100 hours of simulation time (more if real-time) was required to train a 9-DOF mannequin to walk in the simulation. Similarly, 38 days of real-time gameplay was required for Atari 2600 games [2]. The data-hungry nature of these algorithms makes them unsuitable for learning and adaptation in robotics, where the data is much more scarce due to the slow nature of the real physical systems compared to simulated environments. Scarcity of data makes it even more challenging when a robot has to adapt online during its mission because of sudden changes in the dynamics due to component failure (e.g., damages joints), environmental changes (e.g., changes in terrain conditions) or external perturbations (e.g., wind), etc. We refer to these events as different *situations* of the robot.
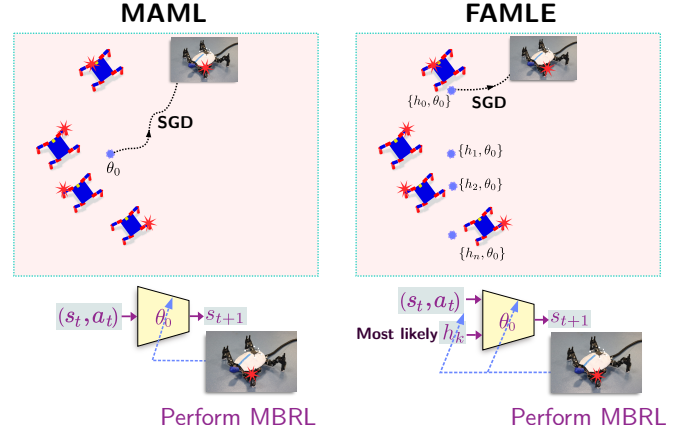
Fig. 1. Basic overview: Compared to model-agnostic meta-learning (MAML) [3], FAMLE meta-learns several initializations for the dynamical model corresponding to various situations (e.g., damage conditions) in the simulation. In FAMLE, this is achieved by using a situation conditioned dynamical model, for which, both the initial situation embeddings ($h_{i=0:n}$), as well as the initial model parameters (θ_0), are jointly meta-trained in such a way that the model can be adapted to similar situations with only a few gradient steps. For model-based RL (MBRL) on the real robot, FAMLE figures out the most suitable embedding out of all the trained embeddings to adapt the model using the real world data.

When data-efficiency is crucial for learning and it is possible to learn a useful dynamical model of the robot from the data, then model-based RL (MBRL) algorithms can be a promising direction [4]. The MBRL algorithms iteratively learn a dynamical model of the robot from the past observations, and using that model as a surrogate of the real robot they either optimize the policy [5]–[7] or a sequence of future actions (as in model predictive control) [8]–[10]. Since MBRL algorithms draw samples from the model instead of the real robot during the policy optimization, these algorithms can be highly data-efficient compared to model-free RL algorithms.

Nevertheless, the data requirement of MBRL algorithms typically scales exponentially with the dimensionality of the input state-action space [4], [11]. As a consequence, for a relatively complex robot, a typical MBRL algorithm still requires a prohibitive interaction time (from several hours to days) to collect enough data to learn a model of dynamics that is good enough for policy optimization [10], [12]. For example, using MBRL approach, an 8-DoF simulated "ant" (actually a quadruped) required around 30 hours of real-time interaction to learn to walk [12]. By contrast, we expect robots to adapt in seconds or, at worst, in minutes when they need to adapt to a new situation [4], [13].

Unlike robots, animals adapt to sudden changes (e.g., broken bones, walking for the first time on a snowy terrain)
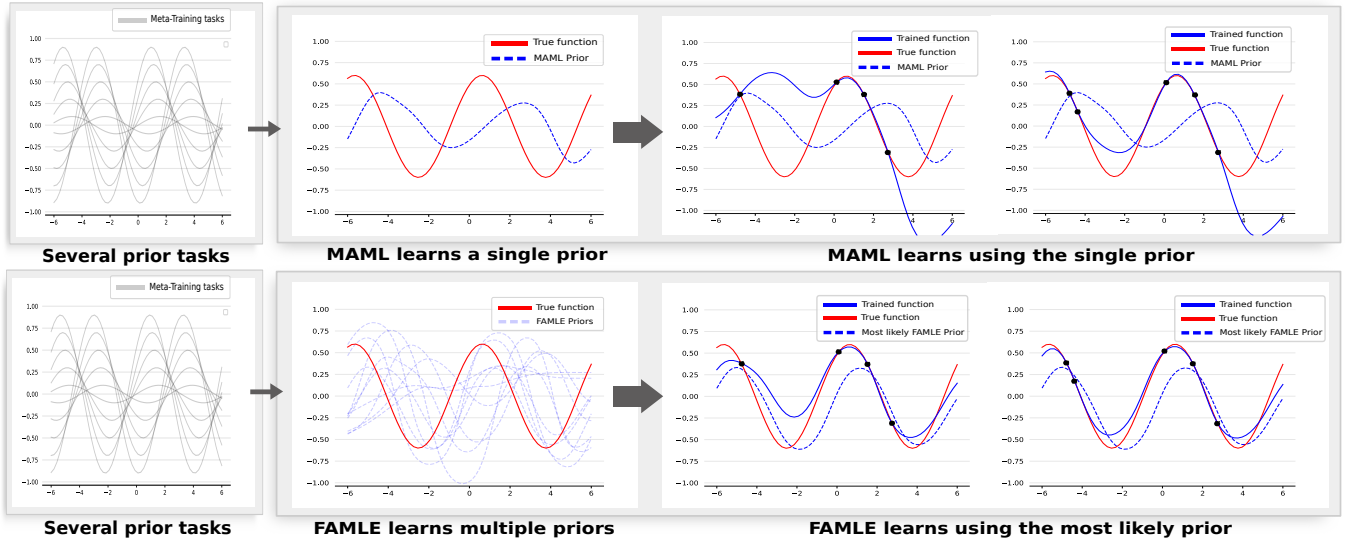
Fig. 2. **Using FAMLE and MAML to learn a simple 1-dimensional sine wave.** Learning several meta-learned priors for the model and selecting the most suitable one for the given data (FAMLE approach) improves data-efficiency. The figure shows that using multiple priors, FAMLE could fit a better function (i.e., closer to the actual function) with 4 and 5 data points than using single priors learned using MAML.

almost immediately. Such a rapid adaptation is possible because animals never learn from scratch; instead, they use their past experience as priors or biases to learn faster. Taking inspiration by this phenomenon, meta-learning approaches use past experiences of the robot in various situations to allow it to learn a new but similar skill or to adapt to a similar situation with only a few observations [3], [9]. When used with MBRL, a meta-learning algorithm such as MAML [3] finds the initial parameters for the dynamical model from where the model can be adapted to similar situations by taking only a few gradient steps. In other words, meta-learning exploits the similarity among the various past experiences to find out a single prior on the initial parameters.

When a robot has to adapt to many different situations (from broken limb to novel terrain conditions), the prior situations used to meta-train the dynamical model can be diverse and without any substantial global similarity among themselves. For example, a 6-legged robot with a broken leg might experience a very different dynamics than the same intact robot walking on rocky terrain. In fact, we observe that when the prior situations are diverse and do not possess a strong global similarity among themselves, using meta-learning to find a single set of initial parameters for the dynamical model is often not enough to learn quickly.

One solution to this problem is to find several initial starting points (i.e., initial model parameters) that are meta-trained in such a way that when the model is initialized with the suitable one, the model can be adapted to the real situation of the robot by performing only a few gradient steps using the past observations. However, the question that arises here is how to meta-train several sets of initial parameters, while still generalizing to situations that were not in the training set. In this work, we propose to achieve this objective by using a single dynamical model that is shared among all the prior situations, but takes an additional input that is

learned so that it corresponds to the situations, which makes it a situation conditioned dynamical model (see Fig. 1 and 2).

To be more precise, our conditional dynamical model not only takes the current-state and action as inputs, but also a d-dimensional vector. This vector is called an embedding of the situation or simply *situation-embedding*. We consider the (initially unknown and randomly set) situation-embeddings as situation-specific parameters of the model, and jointly meta-train all these embeddings as well as the shared model-parameters. In effect, we obtain several meta-trained starting points for the model adaptation – one for each prior training situation (Fig. 2). On the real robot, first, we initialize the model with the meta-trained parameters, then we select the most suitable meta-trained embedding. With the selected embedding as input, we jointly update the embedding as well as the model parameters using gradient descent according to the recent data from the robot. It is to be noted that such joint training of embeddings and model parameters is widely used to learn word embeddings in natural language processing [14].

In summary, our main contribution is an algorithm called FAMLE (Fast Adaptation through Meta-Learning Embeddings) that combines meta-learning and situation embeddings to be able to adapt a dynamical model quickly to a new situation. This model can then be used for model-based RL to optimize its future actions for a given task. FAMLE can be summarized in two steps (Fig. 3):

- **Meta-training:** Generate simulated data for N different situations of the robot, for example, with a broken joint, on rough terrain, on slippery terrain, and so on. Meta-train the model-parameters jointly with the embeddings for each of the simulated situations to have N meta-trained embeddings and one set of meta-train initial model parameters.
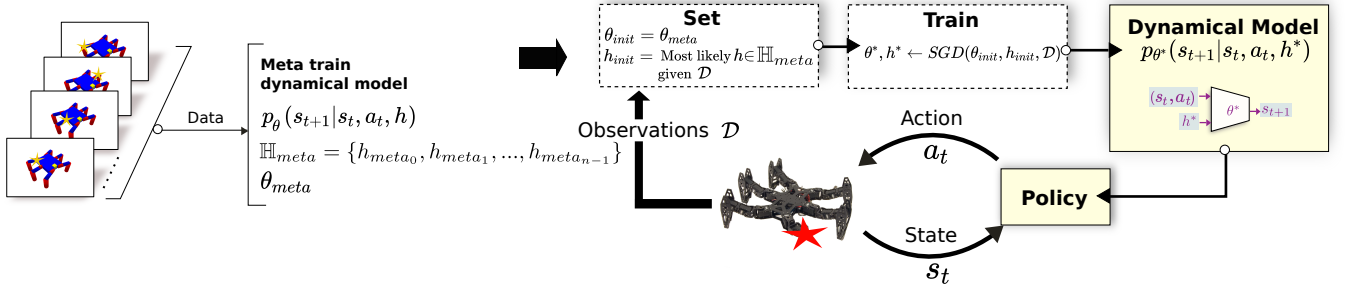
Fig. 3. Overview of FAMLE: first, state-transition data is collected from simulations for n different situations of the robot. Then, the model (i.e., the dynamical model) parameters θ and situation embeddings $\mathbb{H} = \{h_0, ..., h_{n-1}\}$ are meta-trained on the simulated data. On the real robot, when the data is available, FAMLE uses the most suitable embedding from the set $\mathbb{H}$ to adapt the model to the current situation. FAMLE iteratively updates the meta-trained model parameters jointly with the selected embedding using new data and utilizes this model for model predictive control policy.

- **Online adaptation:** On the real robot, at every K step, initialize the model with the meta-trained parameters and the most likely situation embedding out of the N meta-trained embeddings based on the past M observations. Then update the model jointly with the embedding with gradient descent using the past M observations. This model is then used with a model-predictive-control (MPC) policy for K steps. The process repeats until the task is solved.

We compare FAMLE with (a) MBRL with a neural-network dynamics model which was pre-trained with model-agnostic meta-learning (MAML) (b) MBRL with neural-network dynamics model learned from scratch [12], and (c) proximal policy optimization (PPO) [15] on two simulated robots and one real physical quadruped robot. We show that FAMLE allows the robots to adapt to novel damages in significantly fewer time-steps than the baselines. Additionally, we demonstrate that using FAMLE, a physical Minitaur robot (an 8-DoF quadruped) can learn to walk in a minute of interaction in the real world.

II. RELATED WORK

A. Model-based Learning in Robotics

Model-based RL (MBRL) algorithms are some of the most data-efficient learning algorithms in robotics [4]. The core idea of MBRL is to iteratively learn a model of the dynamics of the system and typically use that model either to optimize a policy (called model-based policy search, MBPS) or to optimize the future sequence of action directly (called adaptive model-predictive control, AMPC). In the MBPS framework, many recent algorithms such as PILCO [5], Black-DROPS [6], Multi-Dex [7] have shown promising results towards data-efficient learning in robotics using Gaussian Processes dynamics model. On the other hand, in the AMPC framework, recent works such as [10], [12] used deep neural networks to learn the model of the dynamics of the system. Unlike MBPS, which requires a relatively accurate model of the system dynamics to optimize the policy, AMPC, on the other hand, can use a less precise model of the dynamics as it optimizes the action at every control-step. In order to deal with the model inaccuracies due to small number of samples, the majority of the recent state-of-the-art MBPS algorithms

consider model uncertainties for policy optimization [5]–[7]. Nevertheless, for relatively complex robots, these algorithms still require prohibitively long interaction time (from several hours to days) with the real system when it comes to adapt online to a new situation that perturbs its dynamics, such as damage to joints or a new terrain condition.

B. Using Priors for Data-efficient Learning

In order to accelerate the learning process, many recent work leverage prior knowledge about the system dynamics. In traditional robotics, data efficiency is achieved by simply identifying the tunable parameters of a mathematical model of the system using the observed data from the real robot [16]. In more recent approaches, a parametric or fixed model is "corrected" with a non-parametric model (e.g., Gaussian Processes model) to capture potentially non-linear effects in the dynamics of the system. In particular, model-based policy search algorithm like PILCO [5] or Black-DROPS [6] can be combined with simulated priors and learn to control a cart-pole in 2 to 5 trials [17]–[19]. These approaches learn a "residual model" of the dynamics of the system with Gaussian Processes (GP), i.e., the difference between the simulated and real robot instead of learning the system dynamics from scratch. However, the GP based approaches are limited by the scalability issue of the GP models. Moreover, it is difficult to parameterize different situations that the robot might face in the real world, such as broken legs or faulty actuators, and so on.

Another approach that incorporates priors coming from the simulator to accelerate the learning process is repertoire-based learning. Their key principle is to first learn a large and diverse set of policies in simulation with a "quality diversity" algorithm [20]–[22]. Then use an optimization or search process to pick the policies that works best in the current situation [13], [23]–[26]. Repertoire-based learning algorithm such as APROL [27] and RTE [24] learn how the outcomes of these policies change in the real world compared to the simulated world using GP model, where they use the simulated outcomes as prior mean function to the GP model. In particular, APROL uses several repertoires generated for various situations in the simulation. During the mission, APROL tries to estimate the most suitable repertoire to learn

the policy outcomes for the real robot. APROL has been able to show promising results in fast online adaptation where a damaged hexapod robot with 18 joints could learn to reach a goal location in less than two minutes of interaction despite the damage as well as the reality gap between the simulation and the real world. Approaches like APROL and MLEI [28] show that using the *right prior* out of many priors could significantly improve the data-efficiency in robot learning. However, the repertoire-based learning approaches require expert knowledge to specify the outcome space. Additionally, it is often not possible to learn the dynamics in the outcome space when the outcome strongly depends on the full-state of the system (e.g., a robotic arm where the outcome space is the end-effector position).

Recently, gradient-based meta-learning approaches such as MAML [3] showed a promising direction towards data-efficient learning in robotics using deep neural networks. MAML optimizes the initial parameters for a differentiable model (e.g., a neural-network) of the dynamics such that the model can be adapted to match the actual model of the robot by taking only a few gradient descent steps. In particular, [9] applied MAML to online learning scenarios using model-predictive control with the learned dynamical model. Many recent works on MBRL showed that using a hierarchical dynamical model conditioned on the latent variable gives superior data-efficiency [29]–[32]. In particular, [31], [32] pre-train the dynamical model and the prior distribution over the latent variable for the data gathered from various scenarios. Contrary to these work, we cast our online adaptation problem into "learning-to-learn" framework, similar to MAML [3], where the goal is to learn the initial model-parameters and the initial situation-embeddings using gradient descent in such a way that the model as well as the embeddings can be adapted to unseen situations easily. In effect, our approach produces several initial starting points for the model, each of which can be thought of as a unique *prior* for future adaptation. Selecting the most suitable prior based on the observed data allows us to adapt the model quickly to the real situation.

III. PRELIMINARIES

A. Model-Based Reinforcement Learning

The aim of a RL agent is to maximize the cumulative reward by performing actions in the environment. Formally, a RL problem is represented by a Markov decision process (MDP) which is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0, H)$; where, $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(s'|s,a)$ is the state transition probability for given state s and action a, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, ρ_0 is the initial state distribution, γ is the discount factor, and H is the horizon. An RL agent tries to find a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ to maximize the expected return R given by:

$$R = \mathbb{E}_\pi \Big[\sum_{t=0}^{H-1} r(s,a) \Big] \tag{1}$$

In model-based RL, the above problem is solved by learning the state transition probability function $p(s'|s,a)$ using a

function approximator $p_\theta(s'|s,a)$, which is also called the dynamical model of the system. The model parameter θ is optimized to maximize the log-likelihood of the observed data $\mathcal{D}$ from the environment. This model is then used either to optimize a sequence of actions (as in model-predictive control) or to optimize a policy so that the equation 1 can be maximized.

B. Gradient-based meta-learning

Meta-learning approaches assume that the previous meta-training tasks and the new tasks are drawn from the same task distribution $p(\mathcal{T})$, and these tasks share a common structure or similarity which can be exploited for fast learning. Gradient-based meta-learning such as model-agnostic meta-learning (MAML) [3] tries to find the initial parameters θ for a differentiable parametric model so that taking only a few gradient descent steps from the initial parameters θ produces an effective generalization to the new learning task. More concretely, MAML tries to find an initial set of parameters θ such that for any task $\mathcal{T} \sim p(\mathcal{T})$ with corresponding loss function $\mathcal{L}_\mathcal{T}$, the learner has a low loss after k updates:

$$\min_\theta \mathbb{E}_\mathcal{T} \Big[\mathcal{L}_\mathcal{T}(U_\mathcal{T}^k(\theta)) \Big] \tag{2}$$

where $U_\mathcal{T}^k(\theta)$ is the the update rule (e.g., gradient descent) that updates the parameters θ for k times using the data sampled from $\mathcal{T}$. MAML optimizes this problem with stochastic gradient descent as:

$$\theta := \theta - \alpha \nabla_\theta \mathcal{L}_\mathcal{T}(U_\mathcal{T}^k(\theta)) \tag{3}$$

$$:= \theta - \alpha \nabla_{\tilde{\theta}} \mathcal{L}_\mathcal{T}(\tilde{\theta}) \nabla_\theta U_\mathcal{T}^k(\theta), \text{ where } \tilde{\theta} = U_\mathcal{T}^k(\theta) \tag{4}$$

To ease the computation of equation 4, first-order MAML approximates $U_\mathcal{T}^k(\theta)$ as a constant update of θ as $U_\mathcal{T}^k(\theta) = \theta + \mathbf{b}$ (where $\mathbf{b}$ is a constant). This approximation simplifies equation 4 as:

$$\theta := \theta - \alpha \nabla_{\tilde{\theta}} \mathcal{L}_\mathcal{T}(\tilde{\theta}), \text{ where } \tilde{\theta} = U_\mathcal{T}^k(\theta) = \theta + \mathbf{b} \tag{5}$$

Another first-order meta learning approach called Reptile [33] tries to find a solution θ that is close (in Euclidean distance) to each task $\mathcal{T}$'s manifold of optimal solutions. To achieve this Reptile treats $U_\mathcal{T}^k(\theta) - \theta$ as a gradient which gives the SGD update of θ as:

$$\theta := \theta + \beta (U_\mathcal{T}^k(\theta) - \theta) \tag{6}$$

Unlike MAML, Reptile does not require to split the data into training-set and test-set for meta-learning. In this paper, we use Reptile (Eq. 6) as our meta optimization algorithm due to its computational efficiency and the ease of implementation.

IV. APPROACH

FAMLE involves two steps: (1) meta-learning the situation embeddings and the dynamical model from the data gathered from simulation, and (2) adapting the model as well as the situation-embedding on the real robot for an unseen situation during the mission. In the following subsections, we elaborate these two steps.

A. Meta-learning the situation-embeddings and the dynamical model

We consider a predictive model $p_\theta(s_{t+1}|s_t, a_t, h)$ of the dynamics, where s_{t+1}, s_t, a_t and h are the current-state, the next-state, the applied action and the situation-embedding corresponding to the current situation of the robot. This is represented by a neural network $f_\theta(s_t, a_t, h)$ that predicts the mean of the distribution of next state. In the real world, the robot might face any situation c which comes from a distribution of situations $p(c)$. Since, in this work, we consider a *situation* as any circumstance that perturbs the dynamics of the system, so c represents any unknown dynamics of the system sampled from the distribution of dynamics $p(c)$. To collect the state-transition data from simulation we perform the following steps:

Create empty sets $\mathbb{C}$ and $\mathbb{D}$. Now, for $i = 1$ to N

1) Sample a situation $c_i \sim p(c)$ and insert it in the set $\mathbb{C}$, i.e., $\mathbb{C} = \mathbb{C} \cup \{c_i\}$
2) Instantiate a simulator of the robot for the situation c_i.
3) Perform n random actions on the simulated robot and create data-set $\mathcal{D}_{c_i} = \{(s_t, a_t, s_{t+1})|t = 1, \ldots, n\}$
4) Save the data-set into $\mathbb{D}$, i.e., $\mathbb{D} = \mathbb{D} \cup \{\mathcal{D}_{c_i}\}$

Then, corresponding to each sampled situation $c_{i=1:N}$, we randomly initialize situation-embeddings $\mathbb{H} = \{h_{c_i}|i = 1, \ldots, N\}$. Also, we randomly initialize the model parameter θ. Then the negative log-likelihood loss for any situation $c_i \in \mathbb{C}$ can be written as:

$$\mathcal{L}_{\mathcal{D}_{c_i}}(\theta, h_{c_{i=1:N}}) = \mathbb{E}_{\mathcal{D}_{c_i}}\big[-\log p_\theta(s_{t+1}|s_t, a_t, h_{c_i})\big] \quad (7)$$

Our meta-learning objective is to find initial model-parameters θ_{meta} and situation-embeddings $\mathbb{H}_{meta}$, such that for any situation $c_i \in \mathbb{C}$, performing k gradient descent steps from θ_{meta} and $\mathbb{H}_{meta}$ minimizes the loss given by equation 7. This objective can be written as a meta-optimization problem as:

$$\theta_{meta}, \mathbb{H}_{meta} = \arg\min_{\theta, h_{c_{i=1:N}}} \mathbb{E}_{c \sim \mathbb{C}}\Big[\mathcal{L}_{\mathcal{D}_c}\big(U_c^k(\theta, h_c)\big)\Big] \quad (8)$$

where, $U_c^k(\cdot, \cdot)$ is the gradient descent update rule (applied for k gradient descent steps) that updates the parameters θ and the situation-embedding h_c for any situation $c \in \mathbb{C}$. We optimize the above problem using meta-learning approach similar to Reptile [33] (as in Eq. 6). However, unlike Reptile update in equation 6), we update both θ and embedding h_{c_i} simultaneously. More precisely, at each update step, we randomly choose a situation c_i from the set of situations $\mathbb{C}$ and perform the following update on θ and embedding h_{c_i}:

$$\tilde{\theta}, \tilde{h}_{c_i} = U_{c_i}^k(\theta, h_{c_i}) \quad (9)$$

$$\theta := \theta + \alpha_{meta}(\tilde{\theta} - \theta) \quad (10)$$

$$h_{c_i} := h_{c_i} + \beta_{meta}(\tilde{h}_{c_i} - h_{c_i}) \quad (11)$$

where, α_{meta} and β_{meta} are the meta-learning rate. At convergence, we obtain the meta-trained parameters θ_{meta} and the set of situation-embeddings $\mathbb{H}_{meta}$ for each situation in the set $\mathbb{C}$. Combination of these N situation-embeddings and the meta-trained model-parameters will serve as N different priors for future adaption of the dynamical model to unseen situations.

Algorithm 1 FAMLE: Meta-learning
Require: $\mathbb{D} = \{\}$	▷ Empty set of data-sets
Require: $U_c^k(\cdot, \cdot)$	▷ k steps SGD update rule for situation c

1: **for** $i = 1, 2, \ldots, N$ **do**
2:　　$c_i \sim p(c)$　　　　　　　　　　　　　　　▷ Sample a situation
3:　　$\mathbb{C} \leftarrow c_i$　　　　　　　　　　　　　　　　▷ Save the situation
4:　　$\mathcal{D}_{c_i} = \{(s_t, a_t, s_{t+1})|t = 1, \ldots, n\}$　▷ Simulate and collect data
5:　　$\mathbb{D} = \mathbb{D} \cup \{\mathcal{D}_{c_i}\}$　　　　　▷ Save the data-set for situation c
6: **end for**
7: Randomly Initialize θ　　　　　　　　　　　▷ Model parameters
8: Randomly Initialize $\mathbb{H} = \{h_{c_i}|i = 1, \ldots, N\}$　▷ Situation embeddings
9: **for** $m = 0, 1, \ldots$ **do**
10:　　$\mathcal{D}_{c_i} \sim \mathbb{D}$　　　　　　　　　　　　▷ Sample a data-set
11:　　$\tilde{\theta}, \tilde{h}_{c_i} = U_{c_i}^k(\theta, h_{c_i})$　　　▷ Perform SGD for k steps
12:　　$\theta := \theta + \alpha_{meta}(\tilde{\theta} - \theta)$　　　　▷ Move θ towards $\tilde{\theta}$
13:　　$h_{c_i} := h_{c_i} + \beta_{meta}(\tilde{h}_{c_i} - h_{c_i})$　▷ Move h_{c_i} towards $\tilde{h}_{c_i}$
14: **end for**
15: Return $\theta, \mathbb{H}$　▷ Return meta-trained parameters and embeddings

---

B. Online adaptation to unseen situation

As the robot might face any situation that can perturb its dynamics during the mission, we want to learn a new dynamical model after every K control steps using M recent observations. To learn this model, we set the meta-trained parameters θ_{meta} in the model and compute the likelihood of the M recent observations for each situation embedding in $\mathbb{H}_{meta}$. Then, we use the embedding that maximizes the likelihood of the recent data and train the model parameters as well as the selected embedding. To be more precise, if $\mathcal{D}_M$ is the recent M observations on the robot, then:

$$h_{Likely} = \arg\max_{h \in \mathbb{H}_{meta}} \mathbb{E}_{\mathcal{D}_M}\big[\log p_{\theta_{meta}}(s_{t+1}|s_t, a_t, h)\big] \quad (12)$$

Then we simultaneously update both model parameters $\theta = \theta_{meta}$ and the most likely situation-embedding $h = h_{Likely}$ with by taking k gradient steps:

$$\theta := \theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{D}_M}(\theta, h)$$
$$h_c := h_c - \beta\nabla_{h_c} \mathcal{L}_{\mathcal{D}_M}(\theta, h) \quad (13)$$

After this optimization, we get the optimized model parameters θ^* and situation embedding h^* yielding the model $p_{\theta^*}(s_{t+1}|s_t, a_t, h^*)$. Now using this model, the model predictive control (MPC) method can be used as a policy to maximize the long term reward. In this work, we consider random-sampling shooting [34] to optimize the sequence of action using the model. This method is computationally faster compared to other sampling-based methods such as CEM [35] and relatively easy to implement as well as parallelize. Additionally, due to the randomness, it allows implicit exploration in state-action space, which helps to learn a better model. Random-sampling shooting has been successfully demonstrated as an action sequence optimization method for MPC in recent robot learning papers such as [12]. At any state s, the next action for the robot is optimized as follows:

1) Sample N random trajectories of action where each action is sampled from a uniform distribution: $\{\tau_i|i = 0, \ldots, N-1\}$ and $\tau_i = (a_0^i, a_1^i, \ldots, a_{H-1}^i)$

2) Evaluate the trajectories on the model $f_{\theta^*}(s, a, h^*)$ and select the one that maximizes the total reward.

$$\tau^* = \arg\max_{\tau_i} \sum_{t=0}^{H-1} r(s_t, a_t, f_{\theta^*}(s_t, a_t, h^*)) \quad (14)$$

Where, $r(\cdot, \cdot, \cdot)$ is the reward function.

3) Apply the first action of τ^* and repeat from step (1) until the task is solved.

Pseudo codes for FAMLE is given in Algorithm 1 and Algorithm 2.

Algorithm 2 FAMLE: Fast adaptation & control

Require: $\theta_{meta}, \mathbb{H}_{meta}$ ▷ Meta-learned initial model parameters and embeddings
Require: $\mathcal{D}_M = \phi$ ▷ Empty set for M recent observations
Require: $r(\cdot, \cdot, \cdot)$ ▷ Reward function
1: **while** not $Solved$ **do**
2:     h_{Likely} = most likely $h_{c_i} \in \mathbb{H}_{meta}$ given $\mathcal{D}_M$ and θ_{meta}
3:     $\theta^*, h^* = k$ steps SGD from $\theta_{meta}, h_{Likely}$ using $\mathcal{D}_M$
4:     Apply optimal action $a = MPC(\theta^*, h^*, r(\cdot, \cdot, \cdot))$
5:     $\mathcal{D}_M \leftarrow \mathcal{D}_M \cup \{(s_t, a_t, s_{t+1})\}$ ▷ Insert observation
6:     **if** $size(\mathcal{D}_M) > M$ **then** Remove oldest from $\mathcal{D}_M$
7: **end while**

V. EXPERIMENTAL RESULTS

Here, our goal is to evaluate the data-efficiency of FAMLE and compare it to various baseline algorithms. As a metric for data-efficiency, we focus on the real-world interaction time (or time-steps) required to learn a task by a robot. So, a highly data-efficient algorithm should require fewer time-steps to achieve higher rewards in a reinforcement learning set-up. We compared FAMLE on various tasks against the following baselines and showed that FAMLE requires fewer time-steps to achieve higher rewards than the baselines:

- **PPO:** Proximal Policy Optimization, a model-free policy search algorithm, which is easy to implement, computationally faster, and performs as good as current state-of-the-art model-free policy search algorithms.
- **AMPC:** Adaptive MPC, i.e., MPC using an iteratively learned dynamical model of the system from scratch using past observations with a neural network model.
- **AMPC-MAML:** Adaptive MPC with a meta-trained neural network dynamical model. Here, the network is meta-trained using MAML for the same situations that are used in FAMLE. At test time, the model is updated using the recent data with meta-trained parameters as initial parameters of the network.

For all the MBRL algorithms, we used neural networks that predict the *change in current state* of the robot. To generate the data for the prior situations, we used pybulet physics simulator [36]. The code[1] and the video[2] of the experiments can be found online.

A. Goal reaching with a 5-DoF planar robotic arm

In this simulated experiment, the end-effector of the velocity-controlled (10 Hz) arm has to reach a fixed goal as quickly as possible. The joints of the arm might have

[1]Code: https://github.com/resibots/kaushik_2020_famle
[2]Video: http://tiny.cc/famle_video

various damages/faults: (1) weakened motor (2) wrong voltage polarity, i.e., opposite rotation compared to a normal joint, and (3) dead/blocked motor. Here, the state-space and action space have 12 and 5 dimensions, respectively. The embedding vector size for FAMLE was 5. The dynamical models were learned using neural networks with 2 hidden layers of size 70 and 50. For AMPC-MAML and FAMLE, the dynamical models were meta-trained using the data collected from simulation for 11 different damage situations. For testing, two random damages were introduced on the arm, which were not in the meta-training set. Experiments were performed on 30 replicates, sharing the same test damage condition for all the replicates.
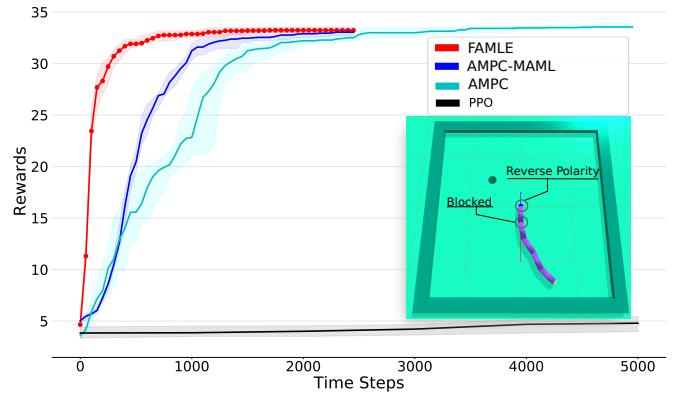

Fig. 4. Goal reaching task: the 5-DoF planer arm has to reach target in minimum number of steps despite damage in its joints. Plot shows median, 25 and 75 percentile of the accumulated reward per episode for 30 replicates. Here, FAMLE maximizes the reward in fewer time-steps than the baselines.

The plot (Fig. 4) shows that FAMLE achieves higher reward much faster than the baselines and solves the task (i.e., reward more than 30) in ∼500 steps (50 seconds). It can be seen that APMC-MAML could not show any significant improvement in the performance compared to AMPC, where the model was learned from scratch. Due to the large variations in the dynamics caused by different damage/fault situations in the meta-training data, meta-trained parameters using MAML could not generalize to all the situations. Thus, it required more data to adapt the model during test time to the current situation. As expected, model-free reinforcement learning baseline PPO could not reach the performance of the model-based approaches within the maximum time-steps limit.

B. Ant locomotion task

In this task, a 4-legged simulated robot (8 joints, torque-controlled, 100 Hz) has to walk in the forward direction as far as possible to maximize the reward. Here, the robot might have (1) blocked joints and/or (2) error in the orientation measurement (i.e., sensor fault). The state-space and the action-space for this problem are 27 and 8 dimensional, respectively. The embedding vector size for FAMLE was 5. The dynamical models were learned using neural networks with 3 hidden layers of size 200, 200 and 100. Meta-training data for AMPC-MAML and FAMLE was collected (applying random actions) from simulations for 20 different

damage/fault situations of the robot. At the beginning of the test, a random joint damage and orientation error (not in the training set) were introduced.
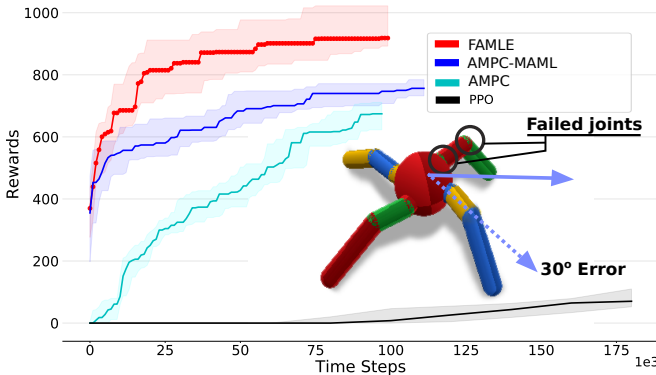

Fig. 5. Ant locomotion task: the damaged ant has to move as far as possible in the forward direction. Plot shows median, 25 and 75 percentile of the accumulated reward per episode for 30 replicates. Here, FAMLE finds higher rewards in much lower time-steps than the baselines.

The plot (Fig. 5) shows that FAMLE achieves higher reward than the baselines and solves the task (i.e., reward more than 800) in ~25000 steps (4.17 minutes of real-time interaction). Similar to the goal-reaching task, here also due to the large variations in the dynamics caused by different situations (especially the orientation faults) in the meta-training data, meta-trained parameters using MAML could not generalize to all the situations and performed worse than FAMLE. As expected, in this experiment also, PPO could not reach the performance of the model-based approaches within the maximum time limit.

C. Quadruped damage recovery

In this online adaptation task, a physical quadruped (12 joints) has to recover from damage to its legs and/or faults in the orientation measurement and reach the goal as quickly as possible. Here, the action is a 4-dimensional vector that modulates the gait of the robot though the period functions associated with each leg[3]. At every second, a new action is applied that produces a new gait on the robot using a low-level controller. The state of the robot includes the 2D position and 2D orientation (sine and cosine of rotation along the vertical axis). The embedding vector size for FAMLE was 20. The dynamical models were learned using neural networks with 2 hidden layers of size 100. The meta-training data for the dynamical model was collected (by applying random actions) from a low fidelity simulator of the robot for total 20 different situations, each of which includes either a joint block or orientation measurement error between 0 to 360 degrees. We tested 3 different situations on the real physical robot: (1) one blocked leg (2) orientation fault, and (3) one blocked leg as well as orientation fault. Additionally, we also evaluated the performance of FAMLE by introducing orientation fault on the robot online (during the deployment).

---

[3]In the preliminary experiments, we were unable to learn a full dynamical model of the physical robots that is accurate enough for MPC.

ORIENTATION FAULT OF 30^0			
	Steps	Time (Sec.)	Success Rate
FAMLE	32.6 ± 2.3	63.4 ± 4.5	100%
AMPC-MAML	61.8 ± 7.2	120.1 ± 13.9	40%
AMPC	-	-	0%
ONE BLOCKED LEG			
	Steps	Time (Sec.)	Success Rate
FAMLE	43.1 ± 6.4	83.8 ± 12.4	100%
AMPC-MAML	78.3 ± 7.2	152.2 ± 14.0	40%
AMPC	-	-	0%
ONE BLOCKED LEG + ORIENTATION FAULT OF 30^0			
	Steps	Time (Sec.)	Success Rate
FAMLE	41.6 ± 7.7	80.9 ± 15.0	100%
AMPC-MAML	79.5 ± 6.1	154.6 ± 11.9	40%
AMPC	-	-	0%

In all the experiments, the goal was 2.5 meters away from the starting position of the robot.
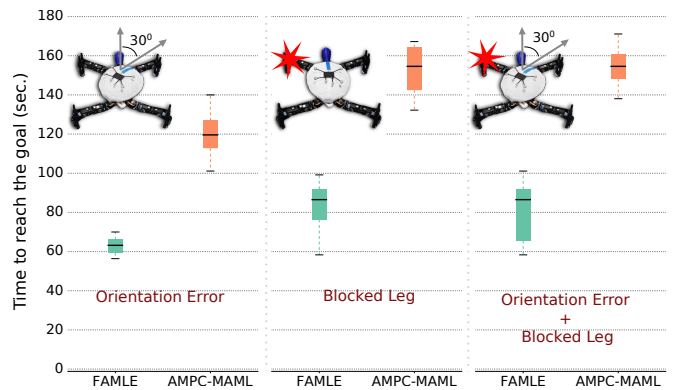

Fig. 6. Quadruped damage recovery task: Box plot (over 10 replicates) shows time required to reach the goal location. Using FAMLE the robot could adapt to both orientation error as well as leg damage and solve the task in less than 2 minutes of interaction.

The table I and box-plot 6 show the comparison of FAMLE with AMPC and AMPC-MAML baselines on the Quadruped damage recovery task. Results show that using FAMLE, the robot could reach the goal 100% of the time by taking significantly less time than the baselines. Using AMPC-MAML, the robot was able to reach the goal only 40% of the time within the maximum allotted time-steps of 80. On the other hand, using AMPC, the robot was never able to reach the goal within the maximum allotted time steps.

D. Minitaur learning to walk

In this experiment, we used the quadruped robot Minitaur from Ghost Robotics. The goal here is to learn to walk in the forward direction as far as possible. Here, the state-space is 6-dimensional (center of mass position and orientation) and action is a 4-dimensional vector (applied at each second) that modulates the gait of the robot through periodic functions associated with each leg (see footnote 3). The embedding vector size for FAMLE was 8. The dynamical models were learned using neural networks with 2 hidden layers of size 20. For meta-training, we collected state transition data from

TABLE II

MINITAUR LEARNING TO WALK

	Max distance	Interaction time
FAMLE	4.8 meters	60 seconds
AMPC-MAML	1.6 meters	90 seconds
AMPC	0.3 meters	130 seconds

the simulator of the robot in pybullet (with random actions) for 3 different friction conditions (default, 0.5 times and 2 times of the default friction) and 3 different weights of the base of the robot (real weight, 1.5 times and 2 times of the real weight).

Due to the high reality gap between the simulated robot and the real robot, a dynamical model trained directly on the data collected from the simulator (for the default weight and friction) performs poorly on the real robot. To verify this, we used such a model on the Minitaur in the simulator as well as on the real robot. On the simulated robot, the robot could immediately walk in the forward direction using model predictive control. However, on the real robot, the robot could not move and failed due to exceeding current limits in the motors. Now, to evaluate the meta-trained model using FAMLE, we used it on the real robot. Thanks to new meta-trained embeddings, the robot could quickly figure out the most suitable embedding to update its model from the real observations. With FAMLE, using only 60 data points (1 minute of real interaction) from the real robot, the robot could consistently walk forward without fail (see Table II).

VI. CONCLUSION

The ability to adapt rapidly to unforeseen situations is one of the main open challenges for robotics. One of the key component to achieve such rapid adaptation is the effective use of prior knowledge. In this work, we have introduced an algorithm called FAMLE and showed how a robot can use prior knowledge from various simulated situations, such as joint damages, sensor error or floor conditions to adapt to various unforeseen situations faster than existing approaches in the literature. We believe, FAMLE is a promising direction towards adaptive robots for long-term missions in the real and uncertain world.

REFERENCES

[1] N. Heess *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.
[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[3] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. of ICML*. JMLR. org, 2017, pp. 1126–1135.
[4] K. Chatzilygeroudis *et al.*, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *arXiv preprint arXiv:1807.02303*, 2018.
[5] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. of ICML*, 2011.
[6] K. Chatzilygeroudis *et al.*, "Black-Box Data-efficient Policy Search for Robotics," in *Proc. of IROS*, 2017.
[7] R. Kaushik, K. Chatzilygeroudis, and J.-B. Mouret, "Multi-objective model-based policy search for data-efficient learning with sparse rewards," in *Conference on Robot Learning*, 2018, pp. 839–855.
[8] G. Williams *et al.*, "Information theoretic mpc for model-based reinforcement learning," in *Proc. of ICRA*, 2017.
[9] A. Nagabandi *et al.*, "Learning to adapt: Meta-learning for model-based control," in *Proc. of ICLR*, 2019.
[10] K. Chua *et al.*, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. of NIPS*, 2018, pp. 4754–4765.
[11] E. Keogh and A. Mueen, "Curse of dimensionality," *Encyclopedia of machine learning*, pp. 257–258, 2010.
[12] A. Nagabandi *et al.*, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. of ICRA*, 2018, pp. 7559–7566.
[13] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.
[14] R. Collobert *et al.*, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.
[15] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[16] J. Hollerbach, W. Khalil, and M. Gautier, *Model Identification*. Cham: Springer International Publishing, 2016, pp. 113–138.
[17] M. Cutler and J. P. How, "Efficient reinforcement learning for robots using informative simulated priors," in *Proc. of ICRA*, 2015.
[18] M. Saveriano, Y. Yin, P. Falco, and D. Lee, "Data-efficient control policy search using residual dynamics learning," *Proc. of IROS*, 2017.
[19] K. Chatzilygeroudis and J.-B. Mouret, "Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics," in *Proc. of ICRA*, 2018.
[20] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arxiv:1504.04909*, 2015.
[21] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.
[22] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Trans. on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2018.
[23] A. Cully and J.-B. Mouret, "Evolving a behavioral repertoire for a walking robot," *Evolutionary Computation*, 2015.
[24] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, "Reset-free trial-and-error learning for robot damage recovery," *Robotics and Autonomous Systems*, vol. 100, pp. 236–250, 2018.
[25] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen, "Evolution of repertoire-based control for robots with complex locomotor systems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 314–328, 2017.
[26] A. Sharma *et al.*, "Dynamics-aware unsupervised skill discovery," in *Workshop at ICLR*, 2019.
[27] R. Kaushik, P. Desreumaux, and J.-B. Mouret, "Adaptive prior selection for repertoire-based online adaptation in robotics," *Frontiers in Robotics and AI*, vol. 6, p. 151, 2020.
[28] R. Pautrat, K. Chatzilygeroudis, and J.-B. Mouret, "Bayesian optimization with automatic prior selection for data-efficient direct policy search," in *Proc. of ICRA*, 2018.
[29] F. Doshi-Velez and G. Konidaris, "Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations," in *Proc. of IJCAI*, vol. 2016. NIH Public Access, 2016, p. 1432.
[30] T. W. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez, "Robust and efficient transfer learning with hidden parameter markov decision processes," in *Proc. of NIPS*, 2017, pp. 6250–6261.
[31] C. F. Perez, F. P. Such, and T. Karaletsos, "Efficient transfer learning and online adaptation with latent variable models for continuous control," *arXiv preprint arXiv:1812.03399*, 2018.
[32] S. Sæmundsson, K. Hofmann, and M. Deisenroth, "Meta reinforcement learning with latent variable gaussian processes," in *Conference on Uncertainty in Artificial Intelligence*, vol. 34, 2018, pp. 642–652.
[33] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.
[34] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.
[35] Z. I. Botev *et al.*, "The cross-entropy method for optimization," in *Handbook of statistics*. Elsevier, 2013, vol. 31, pp. 35–59.
[36] E. Coumans *et al.*, "Bullet physics library," *Open source: bulletphysics. org*, vol. 15, no. 49, p. 5, 2013.