Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL

Anusha Nagabandi
UC Berkeley

Chelsea Finn
UC Berkeley

Sergey Levine
UC Berkeley

1 Introduction

Human and animal learning is characterized not just by a capacity to acquire complex skills, but also the ability to adapt rapidly (Herman, 2017; Flanagan & Wing, 1993) when those skills must be carried out under new or changing conditions. Furthermore, these experiences are remembered, and can be recalled to adapt more quickly when similar disturbances occur in the future (Doyon & Benali, 2005). Since learning entirely new models on such short time-scales is impractical, we desire algorithms that explicitly train models to adapt quickly from small amounts of data, allowing a single model to be maximally useful. Such online adaptation is crucial for intelligent systems operating in the real world, where changing factors and unexpected perturbations are the norm.

In this paper, we propose an algorithm for fast and continuous online learning that utilizes deep neural network models to build and maintain a task distribution, allowing for the natural development of both generalization as well as task specialization. Our working example is continuous adaptation in the model-based reinforcement learning (RL) setting, though our approach generally addresses online learning scenarios with streaming data. We assume each “trial" consists of multiple tasks, and that the delineation between tasks is not provided explicitly – instead, the method must adaptively decide what “tasks" represent, when to instantiate new tasks, and when to continue updating old ones.

We perform adaptation simply by using online stochastic gradient descent (SGD) on model parameters, while maintaining a mixture model over model parameters for different tasks. The mixture is updated via the Chinese restaurant process (CRP) (Stimberg et al., 2012), which enables new tasks to be instantiated as needed over the course of a trial. Although online learning is perhaps one of the oldest applications of SGD (Bottou, 1998), modern parametric models such as deep neural networks are difficult to train online with this method (Sahoo et al., 2017) because they require medium-sized minibatches and multiple epochs to arrive at sensible solutions, which is not suitable for online streaming settings. One of our key observations is that meta-learning can learn a prior initialization for the parameters to make such online adaptation feasible in only a handful of gradient steps.

The meta-training procedure we use is based on model-agnostic meta-learning (MAML) (Finn et al., 2017). Meta-learning and MAML have previously been extended to model-based RL (Clavera et al., 2018), but only for the k-shot adaptation setting: The meta-learned prior model is adapted to the k most recent time steps, but the adaptation is not carried forward in time (i.e., adaptation is always performed from the prior itself). This rigid batch-mode setting is restrictive in an online setup and insufficient for tasks further outside of the training distribution. A more natural formulation is one where the model receives a continuous stream of data and must adapt online to a non-stationary task distribution. This requires fast adaptation and the ability to recall prior tasks, as well as an effective adaptation strategy to interpolate as needed between the two.

The contribution of this paper is an online learning algorithm that uses expectation maximization, in conjunction with a CRP prior on the task distribution, to learn mixtures of neural network models that are updated with SGD. To our knowledge, our work is the first to apply meta-learning to learn streaming online updates. We evaluate our method in the context of model-based RL and show performance improvement over prior methods on a suite of challenging simulated continuous control tasks (unexpected disturbances, environmental changes, and simulated motor failures). Videos available at: `https://sites.google.com/berkeley.edu/onlineviameta`

2 Online Learning with a Mixture of Neural Networks

We formalize our online learning problem setting as follows: at each time step, the model receives an input $\mathbf{x}_t$ and produces a prediction $\hat{\mathbf{y}}_t$. It then receives a ground truth label $\mathbf{y}_t$, which must be used to adapt the model to increase its prediction accuracy on the next input $\mathbf{x}_{t+1}$. The true labels are assumed to come from some task distribution $P(Y_t|X_t, T_t)$, where T_t is the task at time t. We approximate this true model with a predictive model $p_{\theta(T_t)}(\mathbf{y}_t|\mathbf{x}_t)$ on input $\mathbf{x}_t$ for unknown task T_t.

We initialize our task distribution at time step 0 with $\theta_0(T) = \{\theta^*\}$ and $|T| = 1$. As mentioned above, we choose to get this prior parameter vector θ^* from MAML (Finn et al., 2017), which explicitly optimizes for an initialization of a deep network that can achieve fast adaptation. After initializing the task distribution with this prior, we then approximate task identities $\theta_t(T_t)$ and task probabilities $P(T_t)$ using an expectation maximization (EM) algorithm that optimizes the expected log-likelihood:

$$\mathcal{L} = E_{T_t \sim P(T_t|\mathbf{x}_t, \mathbf{y}_t)}[\log p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t)]. \tag{1}$$

2.1 Approximate Online Inference

We use expectation maximization (EM) to update the model parameters. The E step estimates the task distribution $P(T_t)$ at the current time step, and the M step uses these inferred task responsibilities to update the model parameters θ_t into θ_{t+1}.

We first estimate the expectations over all $|T|$ parameters in the task distribution. The posterior of each task probability $P(T_t = T|\mathbf{x}_t, \mathbf{y}_t)$ can be written as follows:

$$P(T_t = T|\mathbf{x}_t, \mathbf{y}_t) \propto p_{\theta(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T)P(T_t = T). \tag{2}$$

We then formulate the task prior $P(T_t)$ using a CRP to enable new tasks to be instantiated during a trial. At time t, the probability of each task T is

$$P(T_t = T) = \frac{n_T}{t - 1 + \alpha} \tag{3}$$

where n_T is the expected number of datapoints in task T for all steps $1, \dots, t-1$, and α is a hyperparameter that controls the instantiation of new tasks. This prior therefore becomes

$$P(T_t = T) = \frac{\sum_{t'=1}^{t-1} P(T_{t'} = T)}{t - 1 + \alpha} \quad \text{and} \quad P(T_t = \text{new}) = \frac{\alpha}{t - 1 + \alpha} \tag{4}$$

Combining the prior and likelihood, we derive the following posterior task probability distribution:

$$P(T_t = T|\mathbf{x}_t, \mathbf{y}_t) \propto p_{\theta(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T)\left[\sum_{t'=1}^{t-1} P(T_{t'} = T) + \delta(T \text{ is new})\alpha\right] \tag{5}$$

Having estimated $P(T_t = T|\mathbf{x}_t, \mathbf{y}_t)$, we next perform the M step, which uses one gradient update to improve the expected log-likelihood in Equation 1 based on the inferred task distributions: . Since each t

$$\theta_{t+1}(T) = \theta^* - \beta \sum_{t'=0}^{t} P_t(T_{t'} = T|\mathbf{x}_{t'}, \mathbf{y}_{t'})\nabla_{\theta_{t'}(T)} \log p_{\theta_{t'}(T)}(\mathbf{y}_{t'}|\mathbf{x}_{t'}) \quad \forall\, T \tag{6}$$

If we assume that $\theta_t(T)$ has already been updated for all previous time steps $t-1, \dots, 0$, we can approximate this update by simply updating the previous parameters $\theta_{t-1}(T)$ on the current sample:

$$\theta_{t+1}(T) = \theta_t(T) - \beta P_t(T_t = T|\mathbf{x}_t, \mathbf{y}_t)\nabla_{\theta_t(T)} \log p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t) \quad \forall\, T \tag{7}$$

This procedure is an approximation, since updates to task parameters $\theta_t(T)$ will in reality change the task probabilities at previous time steps. However, this approximation removes the need to store previously seen data points and yields a fully online, streaming algorithm. We summarize the complete algorithm in Alg. 1.

3 Experiments

We focus on these experimental questions: Can our method 1) autonomously discover some task structure amid a stream of non-stationary data, 2) adapt to tasks that are further outside of the task distribution than can be handled by a k-shot learning approach, 3) recognize and revert to tasks it has seen before, 4) avoid overfitting to a recent task to prevent deterioration of performance upon the next task switch, and 5) outperform other methods?

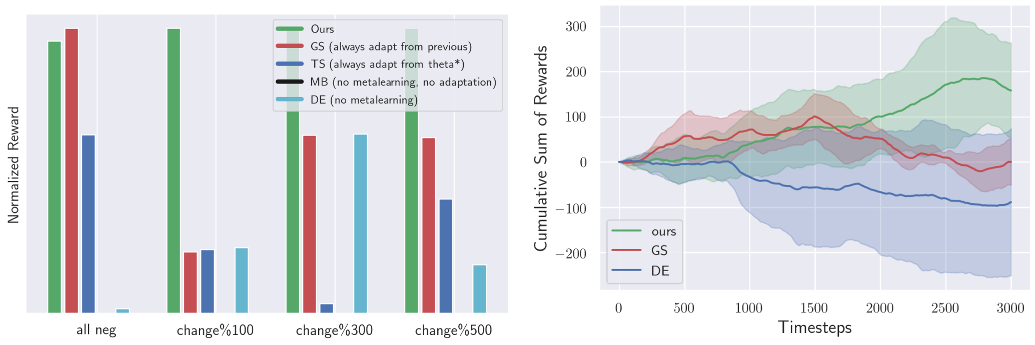

Figure 1: Online motor malfunction trials for half-cheetah, where task distributions either stay constant during a trial or modulate at different frequencies. Online learning is critical for good performance, and continuous gradient steps (GS) leads to overfitting to recent data and forgetting past skills.

To study these questions, we instantiate our online learning via meta-learning method in a model-based RL context. We set the input $\mathbf{x}_t$ to be the concatenation of K previous states and actions $\mathbf{x}_t = [\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_{t-2}, \mathbf{a}_{t-2}, \dots, \mathbf{s}_{t-K}, \mathbf{a}_{t-K}]$, and the output to be corresponding next states $\mathbf{y}_t = [\mathbf{s}_t, \dots, \mathbf{s}_{t-K+1}]$. This provides us with a slightly larger batch of data for each online update, since using individual time steps alone can be very noisy. The underlying predictive model represents each transition as an independent Gaussian, such that $p_\theta(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{K} \mathcal{N}(\mathbf{s}_{t-i+1}; f_\theta(\mathbf{s}_{t-i}, \mathbf{a}_{t-i}), \sigma^2)$, where σ^2 is a constant. To perform control, the model with the highest task probability at the current step is used for planning (Clavera et al., 2018).

Algorithm 1 Online Learning with a Parametric Mixture of Neural Networks

```
Require: $\theta^*$ from meta-training
  Initialize $|T| = 1$, $t = 0$, $\theta_0(T) = \{\theta^*\}$
  for each time step $t$ do
    Calculate $p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T)$ for each $T$
    Calculate $P_t = P_t(T_t = T|\mathbf{x}_t, \mathbf{y}_t)$ for each $T$
    Calculate $\theta_{t+1}$ by adapting from $\theta_t$ for each $T$
    Calculate $\theta_{\text{new}}$ by adapting from $\theta^*$ using $\mathbf{x}_{t-1}, \mathbf{y}_{t-1}$
    if $p_{\theta_{new}(t)} > p_{\theta_t(T)}$ $\forall T$ then
      Add $\theta_{\text{new}}$ to $\theta_{t+1}$
      Recalculate $P_t$ for all $T$, using $\theta_{t+1}$
      Recalculate $\theta_{t+1}$ for all $T$, using updated $P_t$
    end if
    $T^* = \text{argmax}_T(p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T))$
    Select $\theta_{\text{best}}(t) = \theta_{t+1}(T^*)$
    Perform prediction $\hat{\mathbf{y}}_t = p_{\theta_{\text{best}}(t)}(\mathbf{y}_t|\mathbf{x}_t)$
  end for
```

We conduct experiments on the half-cheetah agent and a hexapedal crawler agent in the MuJoCo physics engine (Todorov et al., 2012). We compare to several alternative methods. We consider two other approaches that leverage meta-training: (a) adapt at each time step from the meta-trained prior θ^* (**TS**) as typically done (Clavera et al., 2018), (b) adapt at each time step from the previous parameter (**GS**). The latter version oftens overfits to recently observed tasks, so it should indicate the importance of our method effectively identifying task structure to avoid overfitting and enable recall. We also compare to: **MB** (without meta-training and without adaptation) and **DE** (without meta-training but with adaptation via gradient-descent at each time step (Krause et al., 2017)).

3.1 Half-Cheetah Motor Malfunctions

In the first set of experiments, training data consists of each rollout experiencing a random actuator malfunction (i.e., polarity or magnitude of actions applied to that actuator are altered). Fig. 1 (left) shows that MB and DE are insufficient, and TS is unable to adapt enough to achieve these drastically out-of-distribution test tasks. The continuous adaptation of GS performs well when the task is constant (i.e., set 'sign negative' where all actuators are prescribed to be opposite polarity), but GS deteriorates under a non-stationary task distribution, as in the other tasks. Due to overspecialization on recent incoming data, methods that continuously adapt like GS tend to forget and lose previously existing skills. This overfitting is also illustrated by the deterioration in performance in Fig. 1 (right). Finally, we visualize the latent task probabilities in Fig. 2, where the agent recognizes alternating periods of normal and crippled-leg operation. Note that both recognition and adaptation are done online, using neither a bank of past data nor a human-specified set of task categories.

3.2 Crippling of End Effectors on Six-Legged Crawler

Next, we study another, more complex agent: a 6-legged crawler. In these experiments, all models are trained on random joints being crippled (i.e., unable to apply actuator commands). Fig. 3 (left) shows all methods being comparable for a configuration of crippled joints fixed for the en-

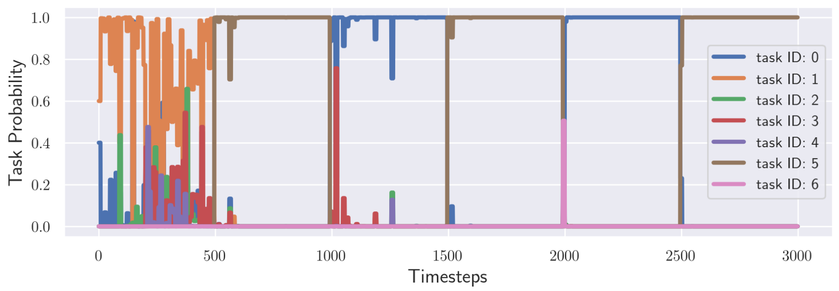

Figure 2: Latent task probability distribution when the underlying motor malfunction changes every 500 time steps. Our method recovers the task structure (online), recognizes task changes, and recalls previously seen tasks.

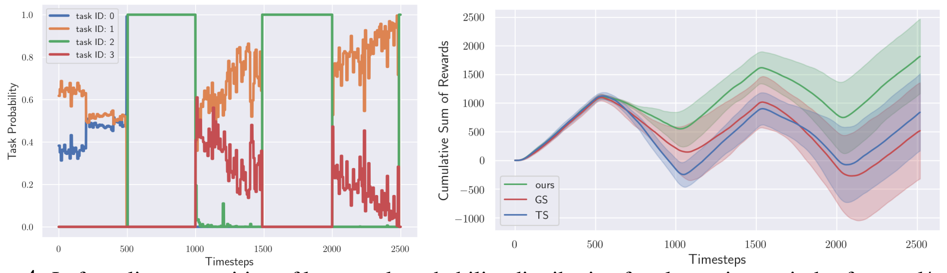

Figure 4: Left: online recognition of latent task probability distribution for alternating periods of normal/crippled experience for crawler. Right: TS/GS not improving after multiple times of seeing the same tasks.

tire duration of its test-time experience. Fig. 3 (right) shows a non-stationary task distribution that illustrates the need for online adaptation (MB fails), the need for a good prior to adapt from (DE fails), the harm of overfitting to recent experience and forgetting older skills (GS low performance), and the need for further adaptation away from the prior (limited TS performance). Fig. 4 (left) shows that our method is able to build its own representation of "task" switches, and we see that this switch does indeed correspond to recognizing regions of leg crippling. Fig. 4 (right) shows the cumulative sum of rewards for trials where 500-1000 and 1500-2000 were periods of crippling. We see that TS does not improve when seeing a task again, GS gets worse, and our method is noticeably better as it sees the task more often. Note that with our method, one skill does not explicitly hinder the other.

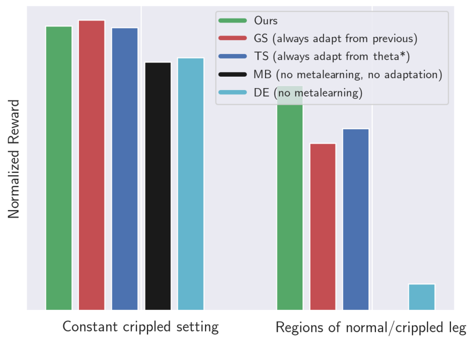

Figure 3: Crawler: end-effector crippling.

We ran another experiment by letting the crawler experience (during each trial) walking straight, making turns, and sometimes having a crippled leg. We compared the performance during the first 500 time steps of "walking forward in a normal configuration" to its last 500 time steps of "walking forward in a normal configuration." While the beginning performance of GS was comparable to our method (average performance difference of +/-10%), its ending performance was **200%** lower. Note the detrimental effect of updating knowledge without allowing for separate task specialization/adaptation.

4 Discussion and Future Work

We presented an online learning method for neural network models that can handle non-stationary, multi-task settings. Our method adapts the model directly with SGD, where an EM algorithm uses a CRP prior to maintain a distribution over tasks. Although SGD generally makes for a poor online learning algorithm in the streaming setting for large parametric models, we observe that, by (1) meta-training the model for fast adaptation with MAML and (2) using our algorithm for probabilistic updates at test time, we can enable effective online learning with neural networks. Our results showed that our method can develop its own notion of task, continuously adapt away from the prior as necessary (for tasks that require more adaptation), and recall tasks it has seen before. Although we use model-based RL as our evaluation domain, our method is general and could be applied to other streaming and online learning settings. An exciting direction for future work would be to apply our method to domains such as time series modeling.

References

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Ignasi Clavera, Anusha Nagabandi, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 2018.

Julien Doyon and Habib Benali. Reorganization and plasticity in the adult brain during learning of motor skills. *Current opinion in neurobiology*, 15(2):161–167, 2005.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning (ICML)*, 2017.

J Randall Flanagan and Alan M Wing. Modulation of grip force with load force during point-to-point arm movements. *Experimental Brain Research*, 95(1):131–143, 1993.

Robert Herman. *Neural control of locomotion*, volume 18. Springer, 2017.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *CoRR*, abs/1709.07432, 2017.

Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017.

Florian Stimberg, Andreas Ruttor, and Manfred Opper. Bayesian inference for change points in dynamical systems with reusable states-a chinese restaurant process approach. In *Artificial Intelligence and Statistics*, pp. 1117–1124, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

5 Other Results

Here, we present a set of experiments whose results demonstrate some unexpected results.

This set of experiments includes a half-cheetah agent traversing terrains of differing slopes. The prior model is meta-trained on data from terrains with random shallow slopes, and the test trials are executed on difficult out-of-distribution tasks such as basins, steep hills, etc. For the three meta-learning and adaptation options (ours, GS, TS), we expect GS to perform poorly due to continuous gradient steps causing it to overfit to recent data; that is, we expect that experience on the upward slopes to lead to deterioration of performance on downward slopes, etc. However, we see (Fig. 5) that the meta-learning procedure seems to have initialized the agent with a parameter space in which these various “tasks" are not seen as substantially different. Thus, even when our method is faced with the option of switching tasks or adding new tasks to its dynamic latent task distribution, it chooses not to do so. Unlike findings that we will see later, it is interesting that the discovered task space here does not correspond to human-distinguishable categorical labels. Finally, we clarify that these tasks of changing slopes are not particularly similar to each other (and that the discovered task space is perhaps useful), because the two non-meta-learning baselines (MB and DE) fail at these test tasks despite having similar performance on the shallow training tasks.

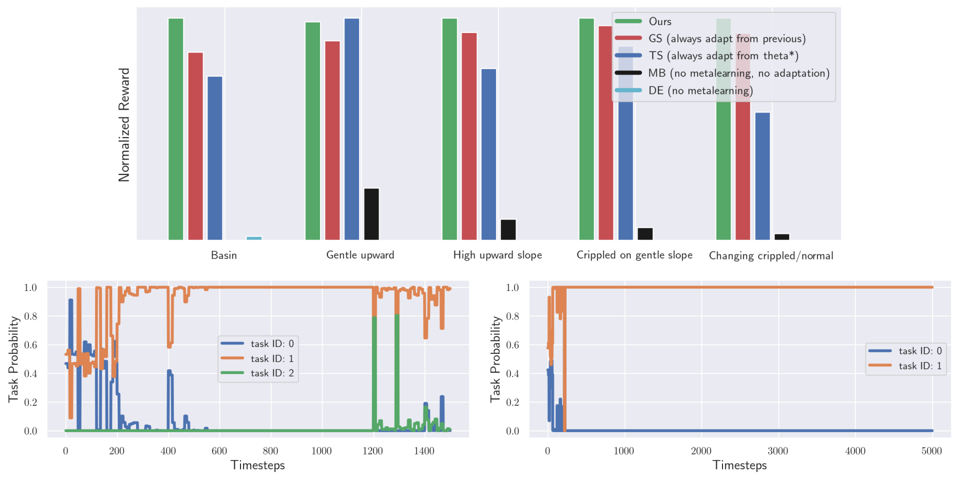

Figure 5: Half-cheetah landscape traversal shows the need for adaptation, similar performance of the three meta-learning approaches, and our method’s choice to only use one latent task variable to describe the varying terrain.

6 Hyperparameters

In all experiments, we use a dynamics model consisting of three hidden layers, each of dimension 500, with ReLU nonlinearities. The control method that we use is random-shooting model predictive control (MPC) where 1000 candidate action sequences each of horizon length H=10 are sampled at each time step, fed through the predictive model, and ranked by their expected reward. The first action step from the highest-scoring candidate action sequence is then executed before the entire planning process repeats again at the next time step.

Below, we list relevant training and testing parameters for the various methods used in our experiments. # Task/itr corresponds to the number of tasks sampled during each iteration of collecting data to train the model, and # TS/itr is the total number of times steps collected during that iteration (sum over all tasks).

Table 1: Hyperparameters for train-time

	Iters	Epochs	# Tasks/itr	# TS/itr	K	outer LR	inner LR (η)
Ours/TS/GS	12	50	16	2000-3000	16	0.001	0.01
MB/DE	12	50	16	2000-3000	16	0.001	N/A

Table 2: Hyperparameters for run-time

	$\alpha(CRP)$	LR (model update)	K (previous data)
Ours	1	0.01	16
GS	N/A	0.01	16
TS	N/A	0.01	16
MB	N/A	N/A	N/A
DE	N/A	0.01	16

7 Test-time Performance vs Training Data

We verify below that as the meta-trained models are trained with more data, their performance on test tasks does improve.

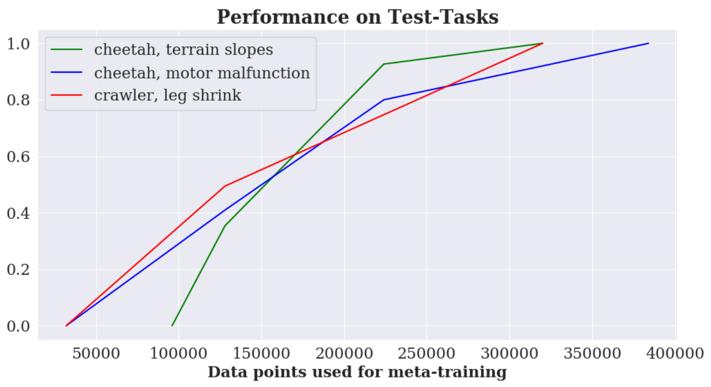

Figure 6: Performance on test tasks (i.e., unseen during training) of models that are meta-trained with differing amounts of data. Performance numbers here are normalized per agent, between 0 and 1.