Curriculum in Gradient-Based Meta-Reinforcement Learning

Bhairav Mehta[*]
Universit de Montral, Mila

Tristan Deleu[†]
Universit de Montral, Mila

Sharath Chandra Raparthy[†]
Mila

Chris J. Pal
Polytechnique Montral, Mila
CIFAR AI Chair

Liam Paull
Universit de Montral, Mila
CIFAR AI Chair

Abstract

Gradient-based meta-learners such as Model-Agnostic Meta-Learning (MAML) have shown strong few-shot performance in supervised and reinforcement learning settings. However, specifically in the case of meta-reinforcement learning (meta-RL), we can show that gradient-based meta-learners are sensitive to task distributions. With the wrong curriculum, agents suffer the effects of *meta-overfitting*, shallow adaptation, and adaptation instability. In this work, we begin by highlighting intriguing failure cases of gradient-based meta-RL and show that task distributions can wildly affect algorithmic outputs, stability, and performance. To address this problem, we leverage insights from recent literature on *domain randomization* and propose meta Active Domain Randomization (meta-ADR), which learns a curriculum of tasks for gradient-based meta-RL in a similar as ADR does for sim2real transfer. We show that this approach induces more stable policies on a variety of simulated locomotion and navigation tasks. We assess in- and out-of-distribution generalization and find that the learned task distributions, even in an unstructured task space, greatly improve the adaptation performance of MAML. Finally, we motivate the need for better benchmarking in meta-RL that prioritizes *generalization* over single-task adaption performance.

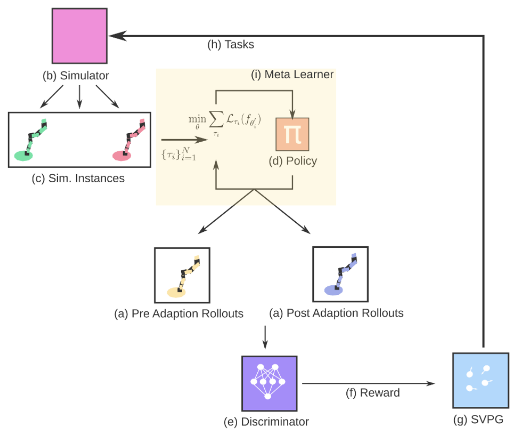

Figure 1: Meta-ADR proposes tasks to a meta-RL agent, helping learn a curriculum of tasks rather than uniformly sampling them from a set distribution. A **discriminator** learns a **reward** as a proxy for task-difficulty, using pre- and post-adaptation rollouts as input. The reward is used to train **SVPG particles**, which find the tasks causing the meta-learner the most difficulty after adaption. The particles propose a diverse set of tasks, trying to find the tasks that are currently causing the agent the most difficulty.

1 Introduction

Meta-learning concerns building models or agents that can learn how to adapt quickly to new tasks from datasets which are orders of magnitudes smaller than their standard supervised learning counterparts. Put differently, meta-learning concerns learning *how to learn*, rather than simply maximizing performance on a single task or dataset. Gradient-based meta-learning has seen a surge of interest, with the foremost algorithm being Model-Agnostic Meta-Learning (MAML) [Finn et al., 2017a]. Gradient-based meta-learners are fully trainable via gradient descent, and have shown strong performance

[*]Correspondence to mehtabha@mila.quebec
[†]Denotes equal contribution

on various supervised and reinforcement learning tasks [Finn et al., 2017a, Rakelly et al., 2019].

The focus of this work is on an understudied hyperparameter within the gradient-based meta-learning framework: the distribution of tasks. In MAML, this distribution is assumed given and is used to sample tasks for meta-training of the MAML agent. In supervised learning, this quantity is relatively well-defined, as we often have a large dataset for a task such as image classification [1]. As a result, the distribution of tasks, $p(\tau)$, is built from random minibatches sampled from this distribution.

However, in the meta-reinforcement learning setting, this task distribution is poorly defined, and is often handcrafted by a human experimenter with the target task in mind. While the task samples themselves are pulled randomly from a range or distribution (i.e a locomoter asked to achieve a target velocity), the distribution *itself* needs to be specified. In practice, the distribution $p(\tau)$ turns out to be an extremely sensitive hyperparameter in Meta-RL: too "wide" of a distribution (i.e the variety of tasks is too large) leads to underfitting, with agents unable to specialize to the given target task even with larger numbers of gradient steps; too "narrow", and we see poor generalization and adaption to even slightly out-of-distribution environments.

Even worse, randomly sampling (as is often the case) from $p(\tau)$ can allow for sampling of tasks that can cause interference and optimization difficulties, especially when tasks are qualitatively different (due to difficulty or task definitions being changed too much by the physical parameters that are varied).

This phenomena, called *meta-overfitting* (or meta-underfitting, in the former, "wide" case), is not new to recent deep reinforcement learning problem settings. Domain randomization [Tobin et al., 2017], a popular *sim2real* transfer method, faces many of the same issues when learning robotic policies purely in simulation. Here, we will show that meta-reinforcement learning has the analogous issues regarding generalization, which we can attribute to the random sampling of tasks. We will then describe the repurposing of a recent algorithm called *Active Domain Randomization* [Mehta et al., 2019], which aims to learn a curriculum of tasks in unstructured task spaces. In this work, we address the problem of meta-overfitting by explicitly optimizing for the task distribution represented by $p(\tau)$. The incorporation of a learned curriculum leads to stronger generalization performance and more robust optimization. Our results highlight the need for continued work in analysis of the effect of task distributions on meta-RL performance and underscoring the potential for curriculum learning techniques.

[1]Even in regression, a function such as a sinusoid is often provided by the experimenter as the task distribution.

2 Background

In this section we briefly cover reinforcement learning, meta-learning, and curriculum learning ideas touched upon in later parts of the paper.

2.1 Reinforcement Learning

We consider a reinforcement learning setting where a task τ is defined as a Markov Decision Process (MDP), a tuple (S, A, T, R, γ) where S is the state space, A is the action space, T is transition function $T : S \times A \to S$, R is a reward function and γ is a *discount factor* which lives within $(0, 1)$. The goal of reinforcement learning is to learn a function π parameterized by θ in such a way that it maximizes the expected total discounted reward [Sutton and Barto, 2018].

2.2 Meta-Learning

Most deep learning models are built to solve only one task, and often lack the ability to generalize and quickly adapt to solve a new set of tasks. Meta-learning involves learning a *learning algorithm* which can adapt quickly rather than learning from scratch. Several methods have been proposed, treating the learning algorithm as a recurrent model capable of remembering past experience [Santoro et al., 2016, Munkhdalai and Yu, 2017, Mishra et al., 2018], as a non-parametric model [Koch et al., 2015, Vinyals et al., 2016, Snell et al., 2017], or as an optimization problem [Ravi and Larochelle, 2017, Finn et al., 2017a]. In this paper, we focus on a popular version of a gradient-based meta-learning algorithm called Model Agnostic Meta Learning (MAML; [Finn et al., 2017a]).

2.2.1 Gradient-based Meta-Learning

The main idea in MAML is to find a good parameter initialization such that the model can adapt to a new task, τ, quickly. Formally, given a distribution of tasks $p(\tau)$ and a loss function $\mathcal{L}_\tau$ corresponding to each task, the aim is to find parameters θ such that the model f_θ can adapt to new tasks with one or few gradient steps. For example, in the case of a single gradient step, the parameters θ'_τ adapted to the task τ are

$$\theta'_\tau = \theta - \alpha \nabla_\theta \mathcal{L}_\tau(\mathcal{D}_{\text{train}}, f_\theta), \tag{1}$$

with step size α, where the loss is evaluated on a (typically small) dataset $\mathcal{D}_{\text{train}}$ of training examples from task

τ. In order to find a good initial value of the parameters θ, the objective function being optimized in MAML is written as

$$\min_{\theta} \sum_{\tau_i} \mathcal{L}_{\tau_i}(\mathcal{D}_{\text{test}}, f_{\theta'_{\tau_i}}), \tag{2}$$

where it evaluates the performance in generalization on some test examples $\mathcal{D}_{\text{test}}$ for task τ. The meta objective function is optimized by gradient descent where the parameters are updated according to

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\tau_i} \mathcal{L}_{\tau_i}(\mathcal{D}_{\text{test}}, f_{\theta'_{\tau_i}}), \tag{3}$$

where β is the outer step size.

2.3 Meta-Reinforcement Learning

In addition to few-shot supervised learning problems, where the number of training examples is small, meta-learning has also been successfully applied to reinforcement learning problems. In meta-reinforcement learning, the goal is to find a policy that can quickly adapt to new environments, generally from only a few trajectories. Rakelly et al. [2019] treat this problem by conditioning the policy on a latent representation of the task, and Duan et al. [2016], Wang et al. [2016] represent the reinforcement learning algorithm as a recurrent network, inspired by the "black-box" meta-learning methods mentioned above. Some meta-learning algorithms can even be adapted to reinforcement learning with minimal changes [Mishra et al., 2018]. In particular, MAML has also shown some success on robotics applications [Finn et al., 2017b]. In the context of reinforcement learning, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ are datasets of trajectories sampled by the policies before and after adaptation (i.e rollouts in $\mathcal{D}_{\text{train}}$ are sampled before the gradient step in Equation 1, whereas those in $\mathcal{D}_{\text{test}}$ are sampled after). The loss function used for the adaptation is REINFORCE [Williams, 1992], and the outer, meta objective in Equation 3 is optimized using TRPO [Schulman et al., 2015].

2.4 Active Domain Randomization

Active Domain Randomization (ADR) [Mehta et al., 2019] builds upon the framework of Domain Randomization [Tobin et al., 2017]. Domain randomization, a useful, zero-shot technique for transferring robot policies from simulation to real hardware, uniformly samples randomized environments (effectively, tasks) that an agent must solve. ADR improves DR by learning an active policy that proposes a curriculum of tasks to train an inner-loop, black-box agent. ADR, mostly used in the zero-shot learning scenario of *simulation-to-real transfer*, uses Stein Variational Policy Gradient (SVPG) [Liu et al., 2017] to learn a set of parameterized particles, $\{\mu_{\phi_i}\}_{i=1}^{N}$ that proposes randomized environments which are subsequently used to train the agent.

SVPG benefits from both the Maximum-Entropy RL framework [Ziebart, 2010] and a kernel term that repulses similar policies to encourage particle, and therefore task, diversity. This allows SVPG to hone in on regions of high reward while maintaining variety, which allows ADR to outperform many existing methods in terms of performance and generalization on zero-shot learning and robotic benchmarks.

To train the particles, ADR uses a discriminator to distinguish between trajectories generated in a proposed randomized environment and those generated by the same policy in a default, reference environment. Intuitively, ADR is optimized to find environments where the same policy produces different behavior in the two types of environments, signalling a probable weakness in the policy when evaluated on those types of randomized environments.

3 Related Work

When discussing meta-reinforcement learning, to the best of our knowledge, the task distribution $p(\tau)$ has never been studied or ablated upon. As most benchmark environments and tasks in meta-RL stem from two papers ([Finn et al., 2017a, Rothfuss et al., 2018], with the task distributions being prescribed with the environments), the discussion in meta-RL papers has almost always centered around the computation of the updates [Rothfuss et al., 2018], practical improvements and approximations made to improve efficiency, or learning exploration policies with meta-learning [Stadie et al., 2018, Gurumurthy et al., 2019, Gupta et al., 2018]. In this section, we briefly discuss prior work in curriculum learning that bears the most similarity to the analyses we conduct here.

Starting with the seminal curriculum learning paper [Bengio et al., 2009], many different proposals to learn an optimal *ordering* over tasks has been studied. Curriculum learning has been tackled with Bayesian Optimization [Tsvetkov et al., 2016], multi-armed bandits [Graves et al., 2017], and evolutionary strategies [Wang et al., 2019] in supervised learning and reinforcement learning settings, but here, we focus on the latter. However, in most work, the task space is almost always discrete, with a teacher agent looking to choose the best next task over a set of N pre-made tasks. The notion of *best* has also been explored in depth, with metrics being based on a variety of things from ground-truth accuracy or reward to adversarial gains between a teacher and student agent [Pinto et al., 2017].

However, up until recently, the notion of continuously-parameterized curriculum learning has been studied less often. Often, continuous-task curriculum learning exploits a notion of *difficulty* in the task itself. In order to get agents to hop over large gaps, it's been empirically easier to get them to jump over smaller ones first [Heess et al., 2017]; likewise, in navigation domains, its been easier to show easier goals and *grow* a goal space [Pong et al., 2019], or even work backwards towards the start state in a reverse curriculum manner [Florensa et al., 2017].

While deep reinforcement learning, particularly in robotics, has a seen a large amount of curriculum learning papers in recent times [Mehta et al., 2019, OpenAI et al., 2019], curriculum learning has not been extensively researched in meta-RL. This may be partly due to the naissance of the field; only recently was a large-scale, multi-task benchmark for meta-RL released [Yu et al., 2019]. As we hope to show in this work, the notions of tasks, task distributions, and curricula in meta-learning are fruitful avenues of study, and can make (or break) many of the meta-learning algorithms in use today.

4 Motivation

We begin with a simple question:

Does the meta-training task distribution in meta-RL really matter?

To answer this question, we run a standard meta-reinforcement learning benchmark, *2D-Navigation-Dense*. In this environment, a point-mass must navigate to a goal, with rewards given at each timestep proportional to the Euclidean distance between the goal and the current position.

We take the hyperparameters and experiment setup from the original MAML work and simply change *the task distribution* from which the 2D goal is *uniformly* sampled. We then show generalization results of the final, meta-learned initial policy *after a single gradient step*. We then track the generalization of the one-step adaptation performance across a wide range of target goals.

In *2D-Navigation-Dense*, the training distribution prescribes goals where each coordinate is traditionally sampled between $[-0.5, 0.5]$ (the second plot in Figure 2) with the agent always beginning at $[0, 0]$. We then evaluate each goal in the grid between $[-2, 2]$ at 0.5 intervals, allowing us to test both in- and out-of-distribution generalization.

We see from Figure 2 an interesting phenomenon, particularly as the training environment shifts away from the

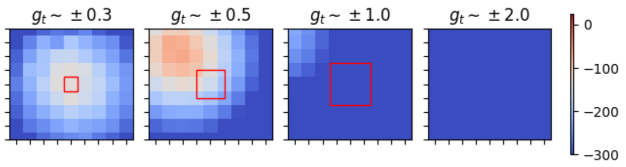

Figure 2: Various agents' final adaption to a range of target tasks. The agents vary only in training task distributions, shown as red overlaid boxes. **Redder is higher reward.**

one which samples goal coordinates $g_t \sim [-0.5, 0.5]$. While the standard environment from [Finn et al., 2017a] generalizes reasonably well, shifting the training distribution even slightly ruins generalization of the adapted policy. What's more, when shown the entire test distribution, MAML fails to generalize to it. We see that on even the simplest environments, the meta-training task distribution seems to have a profound effect, motivating the need for dedicating more attention towards selecting the task distribution $p(\tau)$.

Upon further inspection, we find that shifting the meta-training distribution destabilizes MAML, leading to poor performance when averaged. The first environment, where $g_t \sim [-0.3, 0.3]$ has three out of five random seeds that converge, with the latter two, $g_t \sim [-1.0, 1.0]$ and $g_t \sim [-2.0, 2.0]$, have two and one seeds that converge respectively. The original task distribution sees convergence in all five random seeds tested, hinting at a difference in stability due to the goals, and therefore task distribution, that each agent sees. This hints at a hidden importance of the task distribution $p(\tau)$, a hypothesis we explore in greater detail in the next section.

5 Method

As we saw in the previous section, uniformly sampling tasks from a set task distributions highly affects generalization performance of the resulting meta-learning agent. Consequently, in this work, we optimize for a curriculum over the task distribution $p(\tau)$:

$$\arg\min_{\tau_i \sim p(\tau)} \min_{\theta} \sum_{\tau_i} \mathcal{L}_{\tau_i}(f_{\theta_i'}) \tag{4}$$

where $\theta_i^{'}$ are the updated parameters after a single meta-gradient update.

While curriculum learning has had success in scenarios where task-spaces are structured, learning curricula in unstructured task-spaces, where an intuitive scale of *difficulty* might be lacking, is an understudied topic. However, learning such curricula has seen a surge of interest in the problem of *simulation transfer* in robotics, where policies trained in simulation are transferred zero-shot (no fine-tuning) for use on real hardware. Using a method called domain randomization [Tobin et al., 2017], several recent methods [OpenAI et al., 2019, Mozifian et al., 2019] propose how to learn a curriculum of *randomizations* - which randomized environments would be most useful to show the learning agent in order to make progress on the held-out target task: the real robot.

In the meta-RL setting, the learned curriculum would be over the space of tasks. For example, in *2D-Navigation-Dense*, this would be where goals are sampled, or in *HalfCheetahVelocity*, another popular meta-RL benchmark, the goal velocity the locomotor must achieve.

As learning the curriculum is often treated as a reinforcement learning problem, it requires a reward in order to calculate policy gradients. While many of the methods from the domain randomization literature use proxies such as completion rates or average reward, the optimization scheme depends on the reward function of the *task*. In meta-learning, optimization and reward maximization on a *single* task is not the goal, and such an approach may lead to counter-intuitive results.

A more natural fit in the meta-learning scenario would be to somehow use the *qualitative difference* between the pre- and post-adaptation trajectories. Like a good teacher with a struggling student, the curriculum could shift towards where the meta-learner needs help. For example, tasks in which *negative adaptation* [Deleu and Bengio, 2018] occurs, or where the return from a pre-adapted agent is higher the post-adapted agent, would be prime tasks to focus on for training.

To this end, we modify *Active Domain Randomization* (ADR) to calculate such a score between the two types of trajectories. Rather than using a reference environment as in ADR, we ask a discriminator to differentiate between the pre- and post-adaptation trajectories. If a particular task generates trajectories that can be distinguished by the discriminator after adaptation, we focus more heavily on these tasks by providing the high-level optimizer, parameterized by Stein Variational Policy Gradient, a higher reward.

Concretely, we provide the particles the reward:

$$r_i = \log(f_\psi(y|D_i)) \tag{5}$$

where discriminator f_ψ produces a boolean prediction of whether the trajectory D_i is a pre-adaptation ($y = 0$) or post-adaptation ($y = 1$) trajectory. We present the algorithm, which we term Meta-ADR, in Algorithm 1.

Algorithm 1 Meta-ADR

```
1: **Input** Task distribution $p(\tau)$
2: **Initialize** $\pi_\theta$: agent policy, $\mu_\phi$: SVPG particles, $f_\psi$: discriminator
3: **while not** $max\_epochs$ **do**
4:   **for each** particle $\mu_\phi$ **do**
5:     **sample tasks** $\tau_i \sim \mu_\phi(\cdot)$, bounded by support of $p(\tau)$
6:   **end for**
7:   **for each** $\tau_i$ **do**
8:     $D_{pre}, D_{post}$ = MAML$_{RL}(\pi_\theta, \tau_i)$
9:     Calculate $r_i$ for $\tau_i$ using $D_{post}$ (Eq. (5))
10:  **end for**
11:  *// Gradient Updates*
12:  Update particles using SVPG update rule and $r_i$
13:  Update $f_\psi$ with $D_{pre}$ and $D_{post}$ using SGD.
14: **end while**
```

Meta-ADR learns a curriculum in this unstructured task space without relying on the notion of task performance or reward functions. Note that Meta-ADR runs the original MAML algorithm as a subroutine, but in fact Meta-ADR can run any meta-learning subroutine (i.e Reptile [Nichol et al., 2018], PEARL [Rakelly et al., 2019], or First-Order MAML). In this work, we abstract away the meta-learning subroutine, focusing instead on the effect of task distributions on the learner's generalization capabilities. An advantage of Meta-ADR over the ADR original formulation is that unlike ADR, Meta-ADR requires no additional rollouts, using the rollouts already required by gradient-based meta-reinforcement learners to optimize the curriculum.

6 Results

In this section, we show the results of uniform sampling of the standard MAML agent when changing task distribution $p(\tau)$, while also benchmarking against a MAML agent trained with a learned task distribution using Meta-ADR. All hyperparameters for each task are taken from [Finn et al., 2017a, Rothfuss et al., 2018], with the exception that we take the *final* policy at the end of 200 meta-training epochs instead of the best-performing policy over 500 meta-training epochs. We use the code from [Deleu and Guiroy, 2018] to run all of our experiments. Unless otherwise noted, all experiments are run and averaged across five random seeds. All results are shown after a single gradient step during meta-test time. For each task, we artificially create a generalization range; potentially disjoint from the training distribution of target goals, velocities, headings, etc., and we evaluate each agent both in- and out-of-distribution.

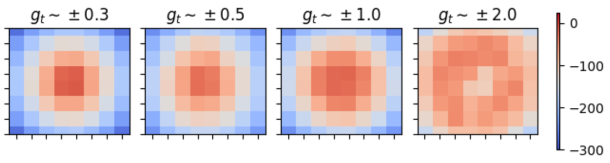

Figure 3: When a curriculum of tasks is learned with Meta-ADR, we see the stability of MAML improve. **Redder is higher reward.**

Importantly, since our focus is on generalization, we **evaluate the final policy**, rather than the standard, *best-performing* policy. As MAML produces a final *initial* policy, when evaluating for generalization for meta-learning, we adapt that initial policy to each target task, and report the adaption results. In addition, in certain sections, we discuss *negative* adaption, which is simply the performance difference between the final, adapted policy and the final, initial policy. When this quantity is negative, as noted in [Deleu and Bengio, 2018], we say that the policy has *negatively* adapted to the task.

We present results from standard Meta-RL benchmarks in Sections 6.1 and 6.2, and in general find that Meta-ADR stabilizes the adaption procedure. However, this finding is not universal, as we note in Section 6.3.

In Subsections 6.3, we highlight a need for better benchmarking and failure cases (overfitting and biased, non-uniform generalization) that both Meta-ADR and uniform-sampling methods seem to suffer from.

6.1 Navigation

In this section we evaluate meta-ADR on two navigation tasks: 2D-Navigation-Dense and Ant-Navigation.

6.1.1 2D-Navigation-Dense

We train the *same* meta-learning agent from Section 4 on *2D-Navigation-Dense*, except this time we use the tasks[2] proposed by Meta-ADR, using a learned curriculum to propose the next best task for the agent. We evaluate generalization across the same scaled up square spanning the ranges of $[-2, 2]$ in both dimensions.

From Figure 3, we see that, with a learned curriculum, the agent generalizes much better, especially *within* the training distribution. A MAML agent trained with a Meta-ADR curriculum also generalizes out-of-distribution with much stronger performance. These results hint at the strong dependence of MAML performance and the task distribution $p(\tau)$, especially when compared to those in Figure 2. Learning such a task curriculum with a method such as Meta-ADR helps alleviate some instability.

[2] In navigation environments, tasks are parameterized by the `x, y` location of the goal.

6.1.2 Ant-Navigation

Interestingly, on a more complex variant of the same task, *Ant-Navigation*, the benefits of such a learned curriculum are minimized. In this task, an eight-legged locomoter is tasked with achieving goal positions sampled from a predetermined range; the standard environment samples the goal positions from a box centered at $(0, 0)$, with each coordinate sampled from $g \sim [-3, 3]$. We systematically evaluate each agent on a grid with both axes ranging between $[-7, 7]$, with a 0.5 step interval.

In Figure 4, we qualitatively see the same generalization across all training task distributions when comparing a randomly sampled task curriculum and a learned one. We hypothesize that this stability comes mainly from the control components of the reward, leading to a smoother, stabler performance across all training distributions. In addition, generalization is unaffected by the choice of distribution, pointing to differences between this task and the simpler version.

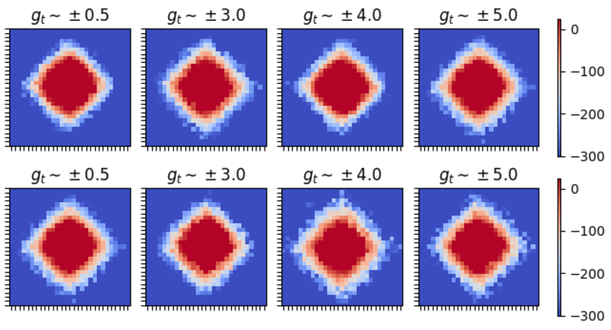

Figure 4: In the Ant-Navigation task, both uniformly sampled goals (top) and a learned curriculum of goals with Meta-ADR (bottom) are stable in performance. We attribute this to the extra components in the reward function. Redder is higher reward.

Compared to the *2D-Navigation-Dense*, *Ant-Navigation* also receives a dense reward related to its distance to target, specifically "control cost, a contact cost, a survival reward, and a penalty equal to its L1 distance to the target position." In comparison, the *2D-Navigation-Dense* task, while a simpler control problem, receives reward information only related to the Euclidean distance to the goal. Counter-intuitively, this simplicity results in *less* stable performance when uniformly sampling tasks, an ablation which we hope to study in future work.

6.2 Locomotion

We now consider *locomotion*, another standard meta-RL benchmark, where we are tasked with training an agent to quickly move in a particular target velocity (Section 6.2.1) or in a particular direction (Section 6.2.2). In this section, we focus on two high-dimensional continuous control problems. In the *AntVelocity*, an eight-legged locomoter must run at a specific speed, with the task space (both for learned and random curricula) being the target velocity. In *Humanoid-Direc-2D*, a benchmark introduced by [Rothfuss et al., 2018], an agent must learn to run in a target direction, θ in a 2D plane.

Both tasks are extremely high-dimensional in both observation and action space. The ant has a (111×1) sized observation space, with each step requiring an action vector of length eight. The Humanoid, which takes in a (376×1) element state, requires an action vector of length 17.

6.2.1 Target Velocity Tasks

When dealing with the target velocity task, we train an ant locomoter to attain target speeds sampled from $v_t \sim [0, 3]$ (Figure 5 left, the standard variant of *AntVelocity*) and speeds sampled from $v_t \sim [0, 5]$ (Figure 5 right).

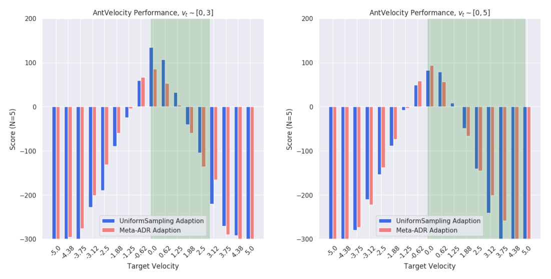

Figure 5: Ant-Velocity sees less of a benefit from curriculum, but performance is greatly affected by a correctly-calibrated task distribution (left). In a miscalibrated one (right), we see that performance from a learned curriculum is slightly more stable.

While we see that learned curricula make insignificant amounts of performance *improvement* over random sampling when shown the same task distribution, we see large differences in performance between task distributions, motivating our hypothesis that $p(\tau)$ is a crucial hyperparameter for successful meta-RL. In additon, we notice that the highest scores are attained on the velocities closer to the easiest variant of the task: a $v_t = 0$, which requires the locomoter to stand completely still. We expand on this oddity in Section 6.3.3.

6.2.2 Humanoid Directional

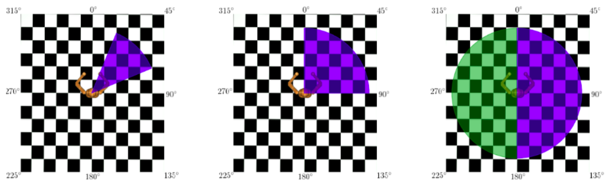

Figure 6: In the high-dimensional Humanoid Directional task, we evaluate many different training distributions to understand the effect of $p(\tau)$ on generalization in difficult continuous control environments. In particular, we focus on *symmetric* variants of tasks - task distributions that mirror each other, such as $0 - \pi$ and $\pi - 2\pi$ in the right panel. Intuitively, when averaged over many trials, such mirrored distrbutions should produce similar trends of in and out-of-distribution generalization.

In the standard variant of *Humanoid-Direc-2D*, a locomoter is tasked with running in a particular direction, sampled from $[0, 2\pi]$. This task makes no distinction regarding target velocity, but rather calculates the reward based on the agent's heading and other control costs.

In this task, we shift the distribution from $[0, 2\pi]$ to subsets of this range, subsequently training and evaluating MAML agents across the entire range of tasks between $[0, 2\pi]$, as seen in the first two panels of Figure 6. Again, we compare agents trained with the standard uniformly-random sampled task distribution against those trained with a learned curriculum using Meta-ADR.

When studying the generalization capabilities on this difficult continuous control task, we are particularly interested in *symmetric* versions of the task; for example, tasks that sample the right and left semi-circles of the task space. We repeat this experiment with many variants of this symmetric task description, and report representative results due to space in Figure 7.

When testing various training distributions, we find that, in general, learned curricula stabilize the algorithm. We see more consistent performance increases, with smaller losses in performance in the directions that *UniformSampling-MAML* outperforms the learned curriculum. However, as noted in Tables 1 and 2, we see that again, the task distribution $p(\tau)$ is an extremely sensitive hyperparameter, causing large shifts in performance when uniformly sampling from those ranges. Worse, this hyperparameter seems to cause *counter-intuitive* gains and drops in performance, both on in *and* out-of-distribution tasks.

While learned curricula seem to help in such a task, a more important consideration from many of these exper-

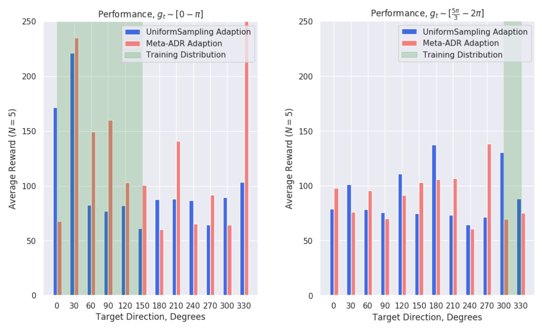

Figure 7: In complex, high-dimensional environments, training task distributions can wildly vary performance. Even in the Humanoid Directional task, Meta-ADR allows MAML to generalize across the range, although it too is affected in terms of total return when compared to the same algorithm trained with "good" task distributions.

Table 1: We compare agents trained with random curricula on different but symmetric task distributions $p(\tau)$. Changing the distribution leads to counter-intuitive drops in performance on tasks both in- and out-of-distribution.

p(τ)	$\theta = 0$	$\theta = 30$	$\theta = 60$
0 - 360	*62.39±7.01*	*64.7±5.38*	*99.93±77.5*
0 - 60	71.51±16.74	82.95±32.06	95.96±37.49
0 - 180	171.77±117.8	221.4±91.66	87.78±45.41
300 - 360	65.64±10.42	95.21±40.08	105.4±50.75
180 - 360	134.52±70.07	79.69±26.01	59.52±2.73

iments is the variance in performance *between* tasks. As *generalization* across evaluation tasks is a difficult metric to characterize due to the inherent issues when *comparing* methods, it is tempting to take the best performing tasks, or average across the whole range. However, as we show in the remaining sections, closer inspection on each of the above experiments sheds light on major issues with the evaluation approaches standard in the meta-RL community today.

Table 2: Evaluating tasks that are *qualitatively* similar, for example running at a heading offset from the starting heading by 30 degrees to the *left or right*, leads to different performances from the same algorithm.

p(τ)	$\theta = 180$	$\theta = 330$	$\theta = 300$
0 - 360	*154.1±121.6*	*81.26±16.39*	*75.47±18.92*
0 - 60	120.2±62.74	84.34±35.9	126.2±101.3
0 - 180	87.78±45.41	103.5±87.24	89.49±31.0
300 - 360	116.03±52.44	81.25±43.82	100.45±48.41
180 - 360	99.1±88.85	80.52±21.79	80.67±21.82

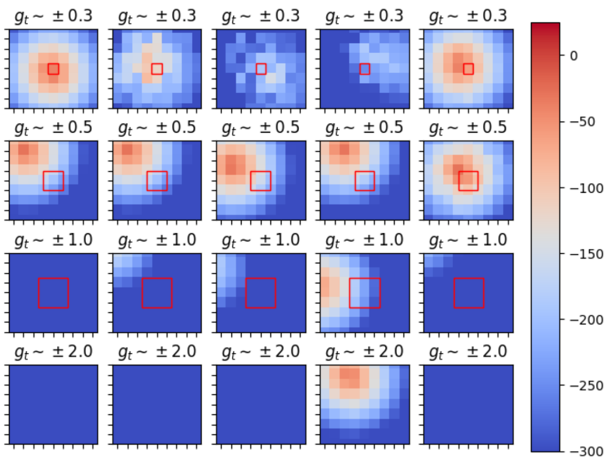

Figure 8: Uniform sampling causes MAML to show bias towards certain tasks, with the effect being compounded with instability when using "bad" task distributions, here shown as $\pm0.3, \pm1.0, \pm2.0$ in the 2D-Navigation-Dense environment.

6.3 Failure Cases of MAML

In this section, we discuss intermediate and auxillary results from each of our previous experiments, highlighting uninterpretable algorithm bias, meta-overfitting, and performance benchmarking in meta-RL.

6.3.1 Non-Uniform Generalization

To readers surprised by the poor generalization capabilities of MAML on such a simple task seen in Figure 2, we offer Figure 8, an unfiltered look at each seed used to calculate each image in Figure 2.

What we immediately notice is the high variance in all but the standard variant of the task, an agent trained on goals with coordinates sampled from $g_t \sim [-0.5, 0.5]$. We even see a reoccurring bias towards certain tasks (visualized as the *top-left* of the grid). Interestingly, when changing the uniform sampling to a learned curriculum, we no longer see such high-variance in convergence across tasks. While our results seem in opposition to many works in the meta-reinforcement learning area, we restate that in our setting, we can only evaluate the *final* policy, as the notion of *best*-performing loses much of its meaning when evaluating for generalization.

6.3.2 Meta-Overfitting

Many works in the meta-reinforcement learning setting focus on final adaption *performance*, but few works focus on the *loss of performance* after the adaption step. Coined by [Deleu and Bengio, 2018] as *negative adaption*, the definition is simple: the loss in performance *af-*

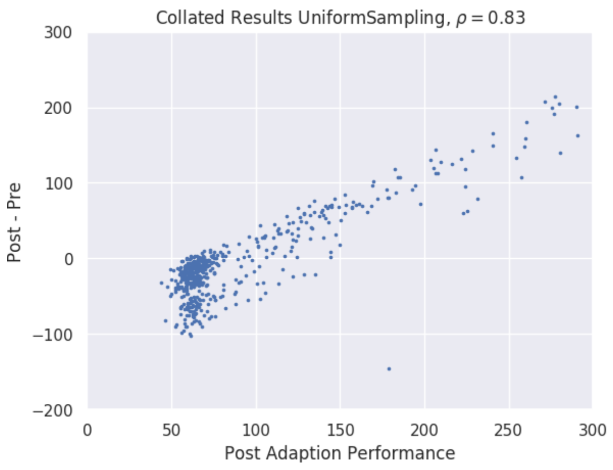

Figure 9: When we correlate the final performance with the amount of quality of *adaption*, we see a troubling trend. MAML seems to overfit to certain tasks, with many tasks that were already neglected during training showing worse post-adaptation returns.

ter a gradient step at meta-test time. Negative adaptation occurs when a pre-adaption policy has overfit to a particular task during meta-training. During meta-test time, an additional gradient steps degrade performance, leading to negative adaptation.

We extensively evaluate negative adaption in the *Humanoid-Direc-2D* benchmark described in Section 6.2.2, providing correlation results between performance and the *difference* between the post- and pre-adaption performance.

When we systematically evaluate negative adaption across all tested *Humanoid-Direc-2D* training distributions, we notice an interesting correlation between performance and the amount of negative-adaptation. Both methods produce near-linear relationships between the two quantities, but when evaluating generalization, we need to focus on the left-hand side of the x-axis, where policies already are performing poorly, and what qualitative effects extra gradient steps have.

We notice a characteristic sign of *meta-overfitting*, where strongly performing policies continue to perform well, but poorly performing ones stagnate, or more often, degrade in performance. When tested, Meta-ADR does not help in this regard, despite having slightly stronger final performance in tasks.

6.3.3 Random Sampling and Performance Improvements

Most alarmingly, we present results on the *evaluation* of tasks, particularly in the Locomotion Velocity tasks. As noted in [Deleu and Bengio, 2018], we see a characteristic *pyramid* when evaluating generalization of locomotion agents trained to achieve a target task. However, in many papers concerning meta-RL, we see monotonic growth curves in performance on such environments. In Figure 10, we show the issues in reporting such curves.

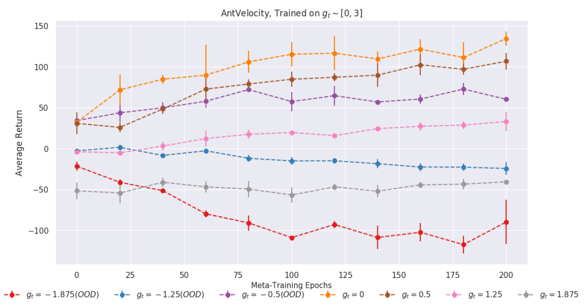

Figure 10: When individually plotting out each target velocity, we see a strong bias towards the easier variants of the task. Only the easier variants of the task produce monotonically increasing learning curves.

When we sample uniformly from the same distribution that we train on, say target velocities pulled from $v_t \sim [0, 3]$, the evaluation sampling can highly affect our results. We evaluate the training curves of the MAML agent, and see that only those closest to the easiest variant of the task ($v_t = 0$, where an agent must stand still) produce monotonically increasing learning curves. More interestingly, target velocities closer to zero but *out-of-distribution* (OOD) show better performance than larger target velocities that are in distribution. As the deviation away from a target velocity of 0 becomes larger, the learning curves stagnate or even start to degrade.

Unlike the previous two sections, which stem interesting research directions such as *why does MAML show bias towards certain tasks* and *how can we fix negative adaption*, our analysis here points towards a fundamental flaw in the *task* design of the Target Velocity Locomotion tasks commonly used in Meta-RL benchmarking.

7 Conclusion

We present *Meta-ADR*, a curriculum learning algorithm suitable for helping gradient-based meta-learners generalize better in a meta-reinforcement learning setting. We show a strong dependence between the performance of MAML and the correct task distribution. When switching out only the random sampling of tasks for such a learned curriculum, we show strong performance across a variety of meta-RL benchmarks. From our experiments, we highlight issues with current meta-RL benchmarking, focusing on a need for *generalization* evalution, a proper, exclusive train-test task separation, and better evaluation tasks in general.

Acknowledgements

The authors gratefully acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds de Recherche Nature et Technologies Quebec (FQRNT), Calcul Quebec, Compute Canada, the Canada Research Chairs, Canadian Institute for Advanced Research (CIFAR) and Nvidia for donating a DGX-1 for computation. BM would like to thank Glen Berseth for helpful discussions in early drafts of this work, and IVADO for financial support.

References

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380.

T. Deleu and Y. Bengio. The effects of negative adaptation in Model-Agnostic Meta-Learning. *CoRR*, abs/1812.02159, 2018. URL `http://arxiv.org/abs/1812.02159`.

T. Deleu and H. S. Guiroy, Simon. Reinforcement Learning with Model-Agnostic Meta-Learning in PyTorch, 2018. URL `https://github.com/tristandeleu/\pytorch-maml-rl/`.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL2: Fast Reinforcement Learning via Slow Reinforcement Learning. 2016. URL `http://arxiv.org/abs/1611.02779`.

C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning*, 2017a. URL `http://arxiv.org/abs/1703.03400`.

C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-Shot Visual Imitation Learning via Meta-Learning. *Conference on Robot Learning*, 2017b. URL `https://arxiv.org/abs/1709.04905`.

C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning, 2017.

A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks, 2017.

A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-reinforcement learning of structured exploration strategies, 2018.

S. Gurumurthy, S. Kumar, and K. Sycara. Mame : Model-agnostic meta-exploration, 2019.

N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver. Emergence of locomotion behaviours in rich environments, 2017.

G. Koch, R. Zemel, and R. Salakhutdinov. Siamese Neural Networks for One-shot Image Recognition. *International Conference on Machine Learning*, 2015.

Y. Liu, P. Ramachandran, Q. Liu, and J. Peng. Stein variational policy gradient, 2017.

B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. *CoRR*, abs/1904.04762, 2019. URL `http://arxiv.org/abs/1904.04762`.

N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A Simple Neural Attentive Meta-Learner. *International Conference on Learning Representations*, 2018. URL `https://arxiv.org/pdf/1707.03141.pdf`.

M. Mozifian, J. C. G. Higuera, D. Meger, and G. Dudek. Learning domain randomization distributions for transfer of locomotion policies. *CoRR*, abs/1906.00410, 2019. URL `http://arxiv.org/abs/1906.00410`.

T. Munkhdalai and H. Yu. Meta Networks. *International Conference on Machine Learning*, 2017. URL `https://arxiv.org/abs/1703.00837`.

A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms, 2018.

OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q.-M. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *ArXiv*, abs/1910.07113, 2019.

L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning, 2017.

V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: State-covering self-supervised reinforcement learning, 2019.

K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. *International Conference on Machine Learning*, 2019. URL `https://arxiv.org/abs/1903.08254`.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. Promp: Proximal meta-policy search, 2018.

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. *International Conference on Machine Learning*, 2016. URL `https://arxiv.org/abs/1605.06065`.

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization, 2015. URL `https://arxiv.org/abs/1502.05477`.

J. Snell, K. Swersky, and R. S. Zemel. Prototypical Networks for Few-shot Learning. *Conference on Neural Information Processing Systems*, 2017. URL `https://arxiv.org/abs/1703.05175`.

B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever. Some considerations on learning to explore via meta-reinforcement learning, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction*. MIT Press, 2018.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017.

Y. Tsvetkov, M. Faruqui, W. Ling, B. MacWhinney, and C. Dyer. Learning the curriculum with Bayesian optimization for task-specific word representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 130–139, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1013. URL `https://www.aclweb.org/anthology/P16-1013`.

O. Vinyals, C. Blundell, T. P. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching Networks for One Shot Learning. *Conference on Neural Information Processing Systems*, 2016. URL `http://arxiv.org/abs/1606.04080`.

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. 2016.

R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions, 2019.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, 1992.

T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.

B. D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, CMU, 2010.