On the Practical Consistency of Meta-Reinforcement Learning Algorithms

Zheng Xiong, Luisa Zintgraf, Jacob Beck, Risto Vuorio, Shimon Whiteson
Department of Computer Science
University of Oxford
zheng.xiong@cs.ox.ac.uk

Abstract

Consistency is the theoretical property of a meta learning algorithm that ensures that, under certain assumptions, it can adapt to any task at test time. An open question is whether and how theoretical consistency translates into practice, in comparison to inconsistent algorithms. In this paper, we empirically investigate this question on a set of representative meta-RL algorithms. We find that theoretically consistent algorithms can indeed usually adapt to out-of-distribution (OOD) tasks, while inconsistent ones cannot, although they can still fail in practice for reasons like poor exploration. We further find that theoretically inconsistent algorithms can be made consistent by continuing to update all agent components on the OOD tasks, and adapt as well or better than originally consistent ones. We conclude that theoretical consistency is indeed a desirable property, and inconsistent meta-RL algorithms can easily be made consistent to enjoy the same benefits.

1 Introduction

For intelligent agents, picking up new tasks is of great practical importance: the real world is noisy and constantly changing, and an agent should reuse its prior knowledge effectively in order to learn new skills. However, the ability of reinforcement learning (RL) agents to do so is still quite limited due to sample inefficiency. Meta-reinforcement learning (meta-RL) algorithms aim to alleviate this problem by meta-training on a set of related tasks to learn a good *inductive bias* on the types of tasks the agent might encounter, which enables fast adaptation to similar (meta-test) tasks.

While a strong inductive bias helps the agent adapt quickly to tasks similar to those seen during meta-training, the agent may be less adaptive to out-of-distribution (OOD) tasks [2]. In fact, it may take an agent longer to adapt than if it had a weaker inductive bias. In a constantly changing world, an agent will likely need to solve OOD tasks that it has not seen before. Consequently, it is desirable for meta-RL algorithms to be able to adapt to OOD tasks beyond their meta-training distribution.

A related concept is *consistency*: a meta learning algorithm is consistent if it can – in theory, under certain assumptions, in the limit – find the optimal solution for any test task [8]. A typical class of consistent meta-RL algorithms are gradient-based methods like MAML [7], E-MAML [18], and ProMP [17]. A typical class of *inconsistent* meta-RL algorithms are context-based methods, such as RL^2 [5, 20], PEARL [16], and VariBAD [25]. The reason gradient-based methods are consistent is because that they adapt using gradient descent at test time, and gradient descent is known to be consistent [23]. On the other hand, the inconsistent context-based methods typically consist of a trajectory encoder and a policy which are meta-trained on a given task distribution via gradient descent, but fixed at meta-test time without further gradient adaptation. Therefore we have no guarantees that the adaptation procedure they learn on a specific task distribution is going to be consistent on arbitrary test task.

5th Workshop on Meta-Learning at NeurIPS 2021, held virtually.

Due to its close relationship to OOD adaptation, consistency is considered a desirable property [8, 13, 14]. However, it remains a *theoretical* property that only holds true under certain assumptions such as infinite adaptation data, or consistency of policy gradients. As these assumptions usually do not hold empirically, it remains an open question how theoretical consistency translates into "practical consistency", i.e., the ability of a meta-RL algorithm to adapt to OOD tasks given a large but limited adaptation budget. In this paper, we aim to shed some light on this question in the context of meta-RL.

We experiment with three representative meta-RL algorithms, MAML [7], RL^2 [5, 20], and VariBAD [25], on a set of diverse tasks, ranging from simple 2D navigation to complex continuous control in Mujoco [19], with different amounts of distribution drift. Our main findings are:

1. In most cases, theoretical consistency provides a good indication of practical consistency, i.e., consistent methods can reasonably adapt to OOD tasks, while inconsistent ones cannot.
2. Theoretically inconsistent algorithms can be made consistent and adapt to OOD tasks if we continue training all agent components on the meta-test tasks via gradient adaptation. Empirically, this strategy achieves similar or even better adaptation performance compared to the originally consistent methods.
3. Theoretical consistency is not always sufficient for practical consistency. We identify one such failure case where the agent fails to solve OOD tasks due to poor exploration.

In conclusion, we believe that theoretical consistency is indeed a desirable property, and inconsistent meta-RL algorithms can easily be made consistent. We also argue that a strong baseline for any study on OOD adaptation is a current state-of-the-art in-distribution meta-RL algorithm which is adapted OOD by continuing to update the meta-learned policy on the meta-test task(s).

2 Background and Related Work

2.1 Formulation of Meta-RL

We define a Markov Decision Process (MDP) as a tuple $M = (\mathcal{S}, \mathcal{A}, R, T, \gamma, H)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $R(r_{t+1}|s_t, a_t, s_{t+1})$ the reward function, $T(s_{t+1}|s_t, a_t)$ the transition function, γ the discount factor, and H the horizon (we omit the initial state transition here for simplicity of notation). The goal of standard RL is to learn a policy π that maximizes the expected return $J(\pi) = \mathbb{E}_{T,\pi}\left[\sum_{t=0}^{H-1} \gamma^t R(r_{t+1}|s_t, a_t, s_{t+1})\right]$.

For meta-RL, we are given a task distribution $p(M)$ over MDPs, where each $M_i \sim p(M)$ is defined as $M_i = (\mathcal{S}, \mathcal{A}, R_i, T_i, \gamma, H)$. The objective is to maximise learning performance in a randomly sampled task with initially unknown reward and transition function.[1] We distinguish between the meta-train distribution p_{train}, and the meta-test distribution p_{test}. In this paper we are interested in adaptation to out-of-distribution (OOD) tasks, i.e., tasks M for which $p_{\text{train}}(M) = 0$ and $p_{\text{test}}(M) > 0$.

2.2 Theoretical Consistency of Meta Learning Algorithms

Finn [8] defines a learning algorithm as a function that processes data $\mathcal{D}$ collected from a task $\mathcal{T}$ to make predictions $y^\star$ from new inputs $x^\star$, and considers a learning algorithm to be consistent if it can find the true function mapping from $x^\star$ to $y^\star$ when provided infinite data, i.e.,

$$\lim_{|\mathcal{D}|\to\infty} f(\mathcal{D}, x^\star) \to y^\star \quad \forall x^\star, y^\star \in \mathcal{T}. \tag{1}$$

In meta-RL, f is often a policy (of which there may be multiple optimal ones) or a value function.

Meta-learning algorithms that adapt via stochastic gradient descent (SGD) are often called consistent [8, 13, 14], citing the empirical evidence that SGD is consistent in practice [23].

We adopt this convention in this paper when categorising algorithms as consistent, but note that deep reinforcement learning algorithms are typically not theoretically consistent. This again motivates our question of whether meta-RL algorithms that are labelled as consistent truly have practical advantages compared to inconsistent methods.

[1] The exact objective may differ between meta-RL methods, so we do not define this formally here. In our experiments, we follow the respective algorithm's default choices.

2.3 Meta-RL Algorithms

Existing meta-RL algorithms can be categorized according to what is meta-learned. We focus on methods that learn adaptive *policies*, i.e., policy networks capable of few-shot adaptation to new tasks, as opposed to meta-learning things like loss functions [10, 12, 15], hyperparameters [21, 22], or environments [9, 3]. The meta-RL algorithms investigated in this paper can be further categorized as gradient-based or context-based. Gradient-based methods [7, 18, 17, 24, 13] adapt to new tasks via gradient updates. These algorithms often differ by network architecture, which parameters are updated, or which loss function is used. Context-based methods [4, 20, 16, 25, 6] aim to infer a latent context per task, and learn a universal policy that conditions on the inferred context. These algorithms often differ by how they encode the context, and how the agent uses this context to act. We experiment with three representative meta-RL algorithms and test their practical consistency:

MAML [7] is a gradient-based method that meta-learns a good model initialization to enable fast adaptation. It adapts to meta-test tasks via policy gradients starting from the meta-learned initialization. MAML is considered to be a consistent method.

RL^2 [5, 20] is a context-based method, where the policy is a recurrent neural network (RNN) that conditions on the state, observed rewards, and taken actions. At test time, no gradients are taken: adaptation happens via the RNN, which has learned to infer the task via its hidden state. RL^2 is considered to be inconsistent, as there is no guarantee that the RNN can encode OOD tasks, or that the RNN's fixed parameters could interpret the OOD hidden state correctly, even with infinite data.

VariBAD [25] is a context-based method. Unlike RL^2, VariBAD decouples learning how to do task inference from policy training. In VariBAD, a VAE [11] is trained to predict environment dynamics given the agent's experience. The VAE's latent state represents the task belief and is passed to the policy. VariBAD is considered to be inconsistent for the same reasons as RL^2.

2.4 Existing Approaches for OOD Adaptation

Although most meta-RL algorithms are designed for in-distribution adaptation, there are also several that explicitly consider OOD adaptation by reusing meta-training data, e.g., via experience relabelling [13] or sample reweighting [6]. MIER [13] only updates the task inference module via gradient descent for in-distribution adaptation, and additionally updates the policy for out-of-distribution adaptation. This is similar to our "continued training via gradient adaptation" (Sec 3.3) setting, which we believe is a strong baseline to consider when studying meta-RL methods for OOD adaptation.

3 Consistency

This paper investigates how theoretical consistency translates into "practical consistency", i.e., the ability of a meta-RL algorithm to adapt to OOD tasks, given a large but limited adaptation budget. Below, we define ways to quantify practical consistency (Sec 3.1), outline settings where consistency matters (Sec 3.2), and propose a way to achieve practical consistency in context-based methods (Sec 3.3).

3.1 Quantifying Practical Consistency

The definition of theoretical consistency (Sec 2.2) in meta-RL assumes that policy gradient methods are consistent with arbitrary parameter initialization, which might not hold true in practice. To investigate whether meta-RL algorithms that are labelled as consistent have a practical advantage, we therefore propose two measures to quantify adaptation performance and speed empirically.

Definition 1 (Consistency Score)**.** For an agent π with initial weights θ_0 and meta-trained weights θ_N, its consistency score c_{score} for frame budget F in an MDP M is defined as

$$c_{\text{score}} = \frac{R_{g(\pi_{\theta_N},F,M)} - R_{\pi_{\theta_0}}}{R_{g(\pi_{\theta_0},F,M)} - R_{\pi_{\theta_0}}}, \tag{2}$$

where R_π is the return of a policy π in the MDP M, and where $g(\pi, F, M) : \Pi \to \Pi$ returns the policy π after training for a given frame budget F on M.

The consistency score aims to measure: When adapting the meta-trained agent on the test task, what is the final performance, and how does this compare to training from scratch?

Definition 2 (Consistency Rate)**.** For an agent π with initial weights θ_0 and meta-trained weights θ_N, its consistency rate $c_{\text{rate}} \in \mathbb{R}_{\geq 0}$ for frame budget F in an MDP M is defined as

$$c_{\text{rate}} = \frac{t(\pi_{\theta_0}, F, M)}{t(\pi_{\theta_N}, F, M)}, \tag{3}$$

with $t(\pi, F, M) : \Pi \to \mathbb{R}_{>0}$, $t(\pi, F, M) = \arg\max R_\pi(f)$, where $R_\pi(f)$ is the return of policy π at training frame f. In words, t computes the number of frames it takes to train a policy π to peak performance within the frame budget F.

The consistency rate aims to measure: How long does it take the meta-trained agent to adapt compared to training from scratch? Assuming that both policies reach the same maximum performance within the adaptation budget, then $c_{\text{rate}} > 1$ implies positive transfer, i.e., the meta-trained policy was able to adapt faster than training from scratch. Conversely, $c_{\text{rate}} < 1$ implies negative transfer. It is important to consider c_{rate} of different algorithms together with their corresponding consistency score c_{score}, because if the policies do not converge to similar maximum performance within the adaptation budget, then c_{rate} alone can be misleading.

3.2 Scenarios for Practical Consistency

We consider two scenarios where measuring consistency is useful in practice.

Single-task setting. Given a model meta-trained in p_{train}, we want to adapt it to a *single* task in p_{test}.

Motivation. The real world is noisy and constantly changing. Meta-learning promises flexible agents than can quickly adapt to such changes, but current methods are often specific to their training task distribution. Overcoming this would have a positive impact for deploying agents in the real world.

Adaptation. For all meta-RL algorithms, we first consider their default adaptation setting for learning on a single new task. For context-based methods, we also consider their consistent variants via gradient adaptation on the test task, as later introduced in Section 3.3.

Evaluation. We compare each meta-RL algorithm to an "expert", trained from scratch on each test task in p_{test}. The practical consistency metrics are computed w.r.t. this expert baseline.

Multi-task Setting. Given a model meta-trained in p_{train}, we want to continue *meta-learning* in p_{test} to be able to quickly adapt to any task in that space.

Motivation. Imagine we have meta-learned a self-driving car that can adapt to different cities in one country, and want to further adapt to different cities in a new country. As there are too many new tasks to solve in p_{test}, it may be more efficient to continue *meta-training* in p_{test} first, instead of adapting to each task separately. Another example is when we first meta-learn in simulation (e.g., a factory robot that stacks things), then continue meta-learning in the real world to bridge the sim2real gap (in a training facility), before deploying the agent (in an actual factory).

Adaptation. For all meta-RL algorithms, we first continue to meta-train the models on multiple tasks from p_{test} following their default meta-training setup, then test the updated model on new tasks from p_{test}. All algorithms follow this same procedure, so we only have three methods to consider here. We call them MAML-CMT, RL^2-CMT and VariBAD-CMT to distinguish the continued multi-task meta-training (CMT) setting here from single-task gradient adaptation under the single-task setting.

Evaluation. We compare against a baseline which meta-learns in p_{test} from scratch. The practical consistency values are computed w.r.t. this in-distribution meta-learning baseline.

For both settings, it is not fair to directly compare the returns between different algorithms, as they each build on different components (e.g., network structure or the RL algorithm used) that can lead to varying absolute performance. The key is to compare different algorithms' practical consistency scores, which are computed w.r.t. their own baselines that use the same components.

3.3 Consistency of Context-based Methods via Gradient Adaptation

As discussed in Section 2.3, context-based methods are inconsistent under their default setting, when trying to adapt to a single OOD test task. We propose a simple gradient adaptation (GA) strategy to

make these methods consistent at test time, i.e., instead of following the fixed meta-learned adaptation procedure, we continue training all agent components with gradient adaptation on the test task.

RL^2-GA. For RL^2, we do gradient updates to the entire RNN weights on the test task. The RNN is still unrolled and reset before each episode. In this way, the only change compared to the default adaptation setting of RL^2 is to update the whole model with gradient adaptation after each episode, but other choices are possible (e.g., freezing the hidden state at some point, or never resetting it).

VariBAD-GA. For VariBAD, we update the VAE and the policy parameters via gradient updates. The RNN in the VAE encoder is still unrolled and reset before each episode.

In our experiments, we contrast these to the standard but inconsistent type of adaption that just unrolls the recurrent networks, i.e., RL^2-default and VariBAD-default.

4 Experiments

In this section we first introduce the tasks domains, and then present the results on the single-task and the multi-task scenario. Detailed experiment settings and more results can be found in the appendix.

4.1 Tasks

2D navigation. The agent starts from the center of an unit circle, and has to navigate to an unknown goal position on the circle's edge. The reward is the negative distance between agent and goal. We divide the circle into four goal spaces (left, right, up, bottom), resulting in 4×4 meta-train-test pairs.

2D sparse navigation. A sparse variant of the above, where the reward is 0 if the agent's distance to the goal is larger than a threshold. We hypothesize that this can prove challenging when adapting to OOD tasks: the agent ought to explore outside the region where it has thus far seen goals. However, it might be overfit to that region and never get to see nonzero rewards from which it can learn.

Cheetah velocity. The goal is to control a half cheetah to run at a specific goal velocity. We consider 6 goal spaces ($[-3, -2]$, $[-2, -1]$, $[-1, 0]$, $[0, 1]$, $[1, 2]$, $[2, 3]$), resulting in 6×6 meta-train-test pairs. Negative velocities mean that the agent has to run backwards. Intuitively, we expect that meta-training could help (e.g., forward $\rightarrow$ forward) or hurt (e.g., forward $\rightarrow$ backward) for OOD adaptation.

Cheetah velocity $\rightarrow$ cheetah direction. The goal of cheetah direction is to control a half cheetah to run either forward or backward as far as possible. By looking at transfer from cheetah velocity to this task, we can investigate adaptation to tasks with a different reward structure.

Cheetah direction $\leftrightarrow$ ant direction. This setting is the hardest, as the two task spaces significantly differ in state space, action space and transition function. We expect there to be minimal positive transfer in this environment, or possibly negative transfer. Ant has a larger state dimensionality, so we pad the observations in Cheetah with zeros. Ant also has a larger action dimension, so we extend the action dimension of cheetah to match that of ant, and only consider the action dimensions with practical meanings when computing the action loss for cheetah.

4.2 Results: Single-Task Setting

Table 1: Average consistency scores of different methods under the single-task setting. The score is averaged over all OOD meta-train-test pairs for each method.

Algorithm	Navi	Sparse-navi	Cheetah-vel	Cheetah-vel $\rightarrow$ dir	Cheeetah-dir $\leftrightarrow$ ant-dir
MAML	0.80	0.00	**0.99**	**0.92**	**0.91**
RL^2-default	-0.49	0.05	0.00	0.22	0.00
RL^2-GA	0.52	**0.42**	0.78	0.80	0.39
VariBAD-default	-0.20	0.03	-0.05	0.27	0.09
VariBAD-GA	**0.91**	0.29	**0.99**	**0.93**	0.66

Table 1 shows the average consistency scores of different algorithms under the single-task setting. We summarize our main findings from different tasks under the single-task setting as follows.

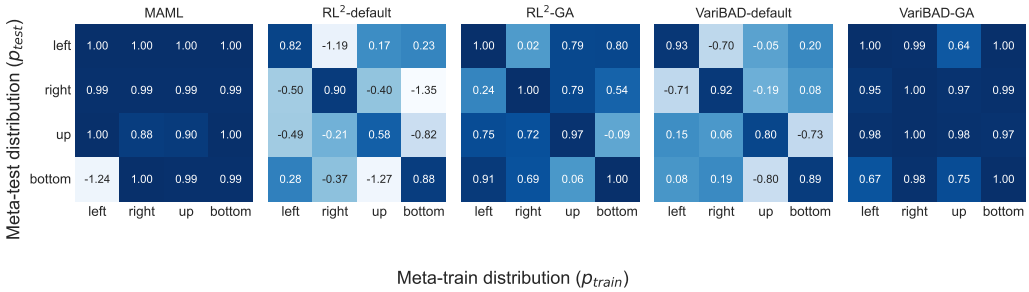

Figure 1: Consistency scores of different meta-RL methods on 2D navigation, in the single-task setting. Darker color means better asymptotic performance. We suggest to look at consistency scores (Fig 1) and consistency rates (Fig 2) together, as in some cases an agent quickly converges to a sub-optimal solution (high consistency rate, low consistency score), which is not desirable in practice.

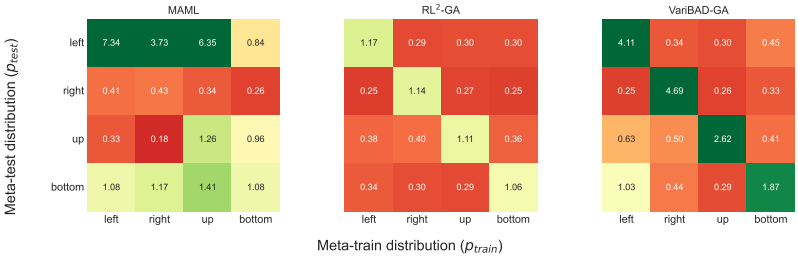

Figure 2: Consistency rates of different meta-RL algorithms on 2D navigation, in the single-task setting. Green represents positive transfer, while red represents negative transfer. We omit RL^2-default and VariBAD-default, as only a few episodes is required for fast adaptation under their default setting, which is much smaller than the adaptation budgets of other methods.

1. The theoretically consistent MAML indeed adapts OOD quite well on most task domains except for sparse navigation. MAML fails on sparse navigation as it only explores within a limited area in the meta-trained direction, and gets no training signal.
2. Context-based methods have low consistency scores, i.e., poor asymptotic performance, under their default setting. Gradient adaptation significantly improves this score, indicating that we can indeed make context-based methods consistent via gradient adaptation. Furthermore, VariBAD-GA achieves performance similar to or even better than the theoretically consistent MAML.
3. Aligned with our intuition, the distribution shifts between p_{train} and p_{test} are correlated with the consistency score and rate, i.e., the smaller the distribution shift, the better and faster a model adapts to OOD tasks. For example, on cheetah velocity, cells closer to the diagonal line usually have higher values (Figure 5). On 2D navigation and sparse navigation, transfer from the opposite direction generally has lower scores than transferring from a neighboring space (Figures 1 and 3).
4. When adapting context-based methods via gradient adaptation, VariBAD-GA performs better than RL^2-GA on most tasks. A possible reason is that VariBAD divides the labour between task encoder and policy, which makes learning easier [25], and might also help OOD adaptation.

We discuss the results of the navigation tasks below. See Appendix B.1 for full results on other tasks.

2D navigation with dense rewards. The consistency score and rate are shown in Figures 1 and 2. Main results: (1) For MAML, most meta-train-test pairs achieve consistency score close to 1, which means OOD adaptation with MAML performs well in general. (2) Context-based methods cannot adapt OOD under their default setting, as illustrated by the low consistency scores in the off-diagonal cells of RL^2-default and VariBAD-default. (3) Gradient adaptation significantly improves the consistency scores of context-based methods, especially for VariBAD. (4) Context-based methods get consistency rate smaller than 1 in most OOD tasks, which indicates negative transfer. MAML is more efficient at OOD adaptation (with both positive and negative transfer).

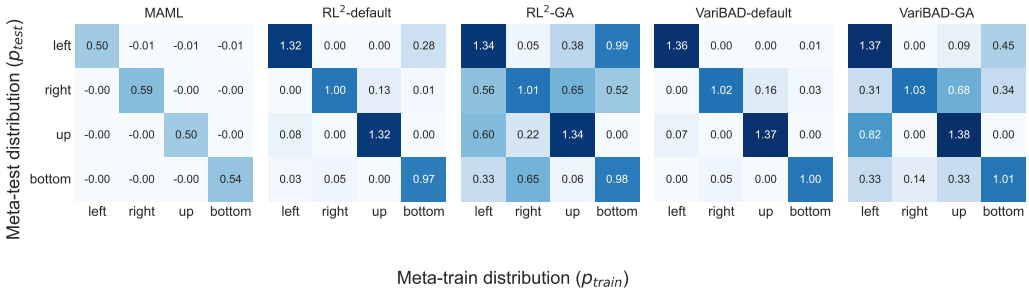

Figure 3: Consistency scores of different meta-RL algorithms on 2D sparse navigation, in the single-task setting.

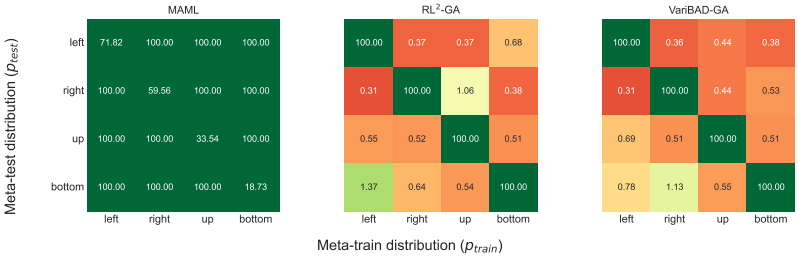

Figure 4: Consistency rates of different meta-RL algorithms on 2D sparse navigation in the single-task setting. MAML has very high consistency rates here because it quickly converges to a poor solution. The maximal rate is 100 since we add a constant of 0.01 to the denominator to avoid division by 0.

2D navigation with sparse rewards. The consistency score and rate are shown in Figures 3 and 4. Main results: (1) In contrast to its consistent performance in the dense-reward setting, MAML fails for OOD adaptation on this sparse navigation task. We believe this is due to poor exploration in regions beyond the meta-training tasks, leading to a lack of learning signal (all rewards are zero). (2) Context-based methods fail under their default setting, but partly solve this task via gradient adaptation. We attribute their advantages over MAML on this task to better exploration introduced by some specific components in their implementations. We discuss this in more detail in Appendix D, where we also visualise the policy's trajectories that show how the meta-learned prior indeed hinders exploration. See also a related discussion on universality by [8].

4.3 Results: Multi-Task Setting

Table 2: Average consistency scores of different methods under the multi-task setting. The score is averaged over all OOD meta-train-test pairs for each method.

Algorithm	Navi	Sparse-navi	Cheetah-vel	Cheetah-vel $\rightarrow$ dir	Cheeetah-dir $\leftrightarrow$ ant-dir
MAML-CMT	0.70	-0.02	0.80	0.73	0.59
RL^2-CMT	0.84	0.71	0.98	1.04	0.35
VariBAD-CMT	**0.99**	**0.79**	**1.05**	**1.06**	**0.66**

Table 2 illustrates the average consistency scores of different algorithms under the multi-task setting. The general trends under the multi-task setting are consistent with what we have observed under the single-task setting, so we refer the reader to the detailed results on each task in Appendix B.2.

One key difference to the single-task setting is that we observe a noticeable increase in the summary score of RL^2 and VariBAD on most tasks. The reason why context-based methods perform better under the multi-task setting may be that the context encoder is trained with more diverse trajectories from different underlying dynamics and thus can learn to better infer the context compared to overfitting to a single environment under the single-task setting.

5 Conclusion and Future Work

Conclusion. In this work, we empirically studied how well different meta-RL algorithms adapt to OOD tasks. We summarize our main findings as follows:

1. Consistent with their theoretical properties, gradient-based methods can adapt to OOD tasks, while context-based methods can't. However, context-based methods adapt as well or better to OOD tasks by simply continuing to train the meta-learned policy via gradient adaptation.

2. Theoretical consistency of meta-RL algorithms may not hold empirically. We identify one such failure case, where the agent explores only in a restricted region due to the meta-learned inductive bias and thus fails to reach OOD goals in navigation tasks with sparse rewards.

3. On most tasks, we observe a correlation between the extent of task distribution shift and adaptation efficiency, i.e., the smaller the distribution shift, the faster the agent adapts in the OOD task space. However, we also observe positive transfer between tasks which are intuitively different (e.g., Cheetah and Ant). This implies a need for accurate quantification of distribution shift, such as those in the transfer learning literature [1].

In summary, we believe that theoretical consistency is indeed a desirable property, and inconsistent algorithms can easily be made consistent. Even with the simple strategy of updating the entire meta-trained agent, the originally inconsistent context-based methods were able to adapt OOD.

Future Work. A drawback of the "gradient adaptation" approach is that it is necessary to explicitly start this process (i.e., know that we are out of distribution). An alternative is to always keep updating everything regardless of whether we are OOD or not. In our experiments however (see Appendix C), we found that with our current setup, this can lead to a slight dip in performance when the test tasks are in-distribution. Future work could focus on how to tackle this instability for in-distribution adaptation or how to automatically detect distribution shifts for adaptation strategy selection.

We also found that insufficient exploration can, unsurprisingly, lead to failure on OOD tasks. Another potential avenue of future work is therefore to develop agents that, when faced with an OOD task, start exploring more widely again.

In this paper, we did not consider meta-RL algorithms that meta-learn aspects other than a policy, such as a loss function, how to set hyperparameters, or training environments. In these methods, inconsistency might be harder to overcome than with context-based methods.

A remaining open question is where the differences in OOD adaptation capabilities of meta-RL algorithms stem from. We conclude this paper by listing in Table 3 some hypotheses about the potential factors that may have positive or negative effects on the learning process. We leave it for future work to develop a better understanding of how exploration, optimization and other issues influence OOD adaptation through model initialization.

Table 3: Hypotheses about the potential positive and negative effects of using a meta-learned model to initialize the learning process on OOD tasks compared to random initialization.

Issue	Positive effects	Negative effects
Exploration	The initial policy saves unnecessary exploration	The initial policy hinders exploration in the right direction
Optimization	Easier to optimize from the initial policy in the parameter space, e.g., fewer local minima in the loss function surface on the path from the initial policy to the solution policy, shorter distance to the solution parameters	Harder to optimize from the initial policy in the parameter space, e.g., the loss function surface of the test task is flat around the initial parameters, greater distance to the solution parameters

Acknowledgments and Disclosure of Funding

Zheng Xiong is supported by UK EPSRC CDT in Autonomous Intelligent Machines and Systems (grant number EP/S024050/1) and AWS. Luisa Zintgraf is supported by the 2017 Microsoft Research PhD Scholarship Program, and the 2020 Microsoft Research EMEA PhD Award. Jacob Beck is supported by the Oxford-Google DeepMind Doctoral Scholarship. Risto Vuorio is supported by EPSRC Doctoral Training Partnership Scholarship and Department of Computer Science Scholarship. This work was supported by a generous equipment grant and a donated DGX-1 from NVIDIA.

References

[1] Ammar, H. B., Eaton, E., Taylor, M. E., Mocanu, D. C., Driessens, K., Weiss, G., and Tuyls, K. (2014). An automated measure of mdp similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[2] Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., and Hassabis, D. (2019). Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422.

[3] Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. (2020). Emergent complexity and zero-shot transfer via unsupervised environment design. *arXiv preprint arXiv:2012.02096*.

[4] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016a). Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR.

[5] Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016b). Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

[6] Fakoor, R., Chaudhari, P., Soatto, S., and Smola, A. J. (2020). Meta-q-learning. In *International Conference on Learning Representations*.

[7] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

[8] Finn, C. B. (2018). *Learning to learn with gradients*. University of California, Berkeley.

[9] Gupta, A., Eysenbach, B., Finn, C., and Levine, S. (2018). Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*.

[10] Houthooft, R., Chen, R. Y., Isola, P., Stadie, B. C., Wolski, F., Ho, J., and Abbeel, P. (2018). Evolved policy gradients. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5405–5414.

[11] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *stat*, 1050:10.

[12] Kirsch, L., van Steenkiste, S., and Schmidhuber, J. (2019). Improving generalization in meta reinforcement learning using learned objectives. In *International Conference on Learning Representations*.

[13] Mendonca, R., Geng, X., Finn, C., and Levine, S. (2020). Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*.

[14] Mitchell, E., Rafailov, R., Peng, X. B., Levine, S., and Finn, C. (2021). Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pages 7780–7791. PMLR.

[15] Oh, J., Hessel, M., Czarnecki, W. M., Xu, Z., van Hasselt, H., Singh, S., and Silver, D. (2020). Discovering reinforcement learning algorithms. *arXiv preprint arXiv:2007.08794*.

[16] Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR.

[17] Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. (2018). Promp: Proximal meta-policy search. In *International Conference on Learning Representations*.

[18] Stadie, B. C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. (2018). Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*.

[19] Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

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

[21] Xu, Z., van Hasselt, H., and Silver, D. (2018). Meta-gradient reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2402–2413.

[22] Zahavy, T., Xu, Z., Veeriah, V., Hessel, M., Oh, J., van Hasselt, H., Silver, D., and Singh, S. (2020). A self-tuning actor-critic algorithm. *arXiv preprint arXiv:2002.12928*.

[23] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization (2016). *arXiv preprint arXiv:1611.03530*.

[24] Zintgraf, L., Shiarli, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR.

[25] Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. (2020). Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations (ICLR)*.

A Experimental Setup

The episode length is set to 100 for navigation, and 200 for Mujoco. For all experiments, results are averaged over 3 seeds. The learning budgets are shown in Table 4. For MAML, we set the batch size of both outer and inner loop to 10, tune the learning rate of inner loop in $\{0.1, 0.05, 0.02, 0.01\}$, and follow the default setting in the PyTorch implementation of MAML[2] for other configurations. For RL^2 and VariBAD, we follow the default setting in the official code of VariBAD[3]. As the meta-training process is unstable in some cases (especially for RL^2), we use the model with the highest evaluation returns during meta-training, instead of the final model after meta-training, for adaptation.

Table 4: Frame number for each task and different learning stages.

Task	Meta-train	Single-task adaptation	Multi-task adaptation
Navigation	10M (MAML) 5M (RL^2, VariBAD)	2M	10M (MAML) 5M (RL^2, VariBAD)
Sparse navigation	20M	5M	20M
Cheetah velocity	10M	2M	10M
Cheetah vel $\rightarrow$ dir	10M	5M	10M
Cheetah dir $\leftrightarrow$ ant dir	20M	10M	20M

Empirically, we compute the consistency score by first smoothing the learning curve with a sliding window of length 5, then using the highest return of the smoothed learning curve as R_π. For the consistency rate, we compute $t(\pi, F, M)$ as the frame number the agent needs to reach a return of $0.98R_\pi$ on the smoothed learning curve (as the maximal return R_π may be achieved long after convergence due to small fluctuation in the learning curve).

B Further Experimental Results

B.1 Single-Task Setting

Cheetah velocity. The consistency score and rate are shown in Figures 5 and 6. Main results: (1) Under their default setting, context-based methods can only partly solve OOD tasks with small shifts in goal velocity, and their performance significantly improves via gradient adaptation. (2) MAML, RL^2-GA and VariBAD-GA show positive OOD adaptation, i.e., similar convergence performance and better learning efficiency compared to the "expert" baseline, if the distribution shift is not too large. Specifically, if we use the median value of each velocity interval to represent the corresponding task space, and use 0 to represent the "expert", then there is a general trend that OOD adaptation is more efficient than single-task learning from scratch if $|b - a| < |b - 0|$, where a and b are the numbers representing the meta-training and meta-test space respectively. (3) For the context-based methods via gradient adaptation, VariBAD generally outperforms RL^2 w.r.t. both adaptation efficiency and final performance, especially when the distribution shift is large.

Cheetah velocity $\rightarrow$ cheetah direction. The consistency score and rate are shown in Figures 7 and 8. Main results: (1) Consistent with the results on other tasks, gradient adaptation significantly improves the performance of context-based methods. (2) Surprisingly, an agent trained to run at a specific velocity in one direction can transfer quite well to the task of running as far as possible in the opposite direction. One possible reason might be that although the goal direction is opposite, the agent can still learn some useful prior knowledge from meta-training, such as how to run (fast), which is essential for solving the test task.

Cheetah direction $\leftrightarrow$ ant direction. The consistency score and rate are shown in Figures 9 and 10. Main results: (1) Gradient adaptation significantly improves the performance of context-based methods. (2) The transfer between the two tasks is asymmetric, i.e., ant direction $\rightarrow$ cheetah direction performs much better than cheetah direction $\rightarrow$ ant direction. The main reason may be that the agent observes additional inputs and has more action outputs when transferring from cheetah to ant.

[2] `https://github.com/tristandeleu/pytorch-maml-rl`

[3] `https://github.com/lmzintgraf/varibad`

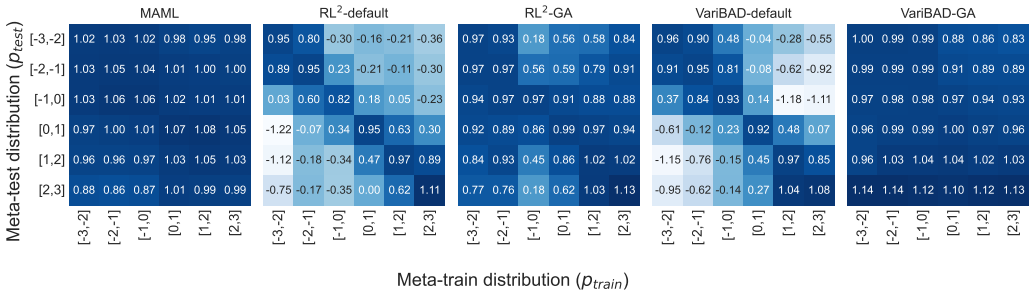

Figure 5: Consistency scores of different meta-RL methods on cheetah velocity; single-task setting.

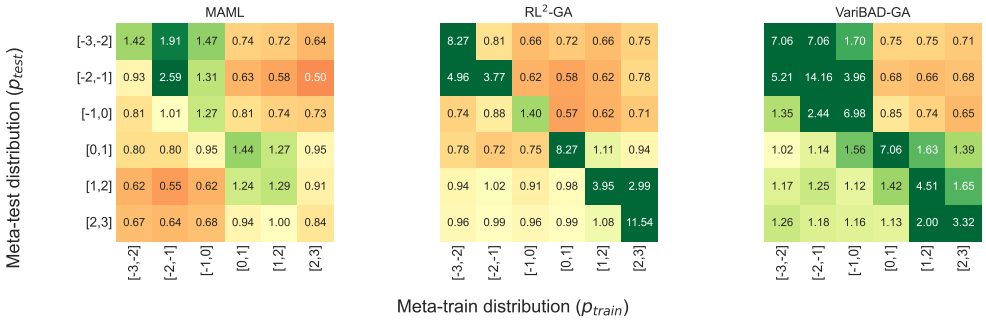

Figure 6: Consistency rates of different meta-RL methods on cheetah velocity; single-task setting.

B.2 Multi-Task Setting

2D navigation with dense rewards. The consistency score and rate are shown in Figures 11 and 12. Main results: (1) W.r.t. asymptotic performance, VariBAD achieves OOD adaptation performance very similar to in-distribution learning, while MAML and RL^2 are worse when the meta-test goal direction is opposite to that of meta-training. (2) W.r.t. learning efficiency, none of the algorithms can adapt to the OOD space faster than in-distribution meta-learning, as shown by the fact that no off-diagonal cell has consistency rate larger than 1.

2D navigation with sparse rewards. The consistency score and rate are shown in Figures 13 and 14. Main results: (1) MAML totally fails with OOD adaptation due to poor exploration, similar to its result on the same task under the single-task setting. (2) RL^2 and VariBAD in some cases can solve this challenging OOD adaptation task with sparse rewards, while the task also becomes much harder for them if the model is transferred to the opposite direction.

Cheetah velocity. The consistency score and rate are shown in Figures 15 and 16. Main results: (1) W.r.t. consistency score, MAML's performance drops as the distribution shift increases, while RL^2 and VariBAD are generally more robust (expect for several meta-train-test pairs of RL^2). (2) W.r.t. consistency rate, generally all methods adapt to OOD tasks more efficiently than in-distribution learning if the distribution shift is small, while RL^2 and VariBAD better maintain their high adaptation efficiency as distribution shift increases compared to MAML.

Cheetah velocity $\rightarrow$ cheetah direction. The consistency score and rate are shown in Figures 17 and 18. Main results: (1) For MAML, the OOD learning performance is slightly lower than in-distribution learning. (2) For RL^2 and VariBAD, although the meta-trained models are only capable of running in one direction within a specific velocity range, they adapt quite well to the new task of running in both directions, and achieve similar learning efficiency compared to in-distribution meta-learning from scratch.

Cheetah direction $\leftrightarrow$ ant direction. The consistency score and rate are shown in Figures 19 and 20. The transfer is asymmetric between the two tasks, similar to the trend under the single-task setting.

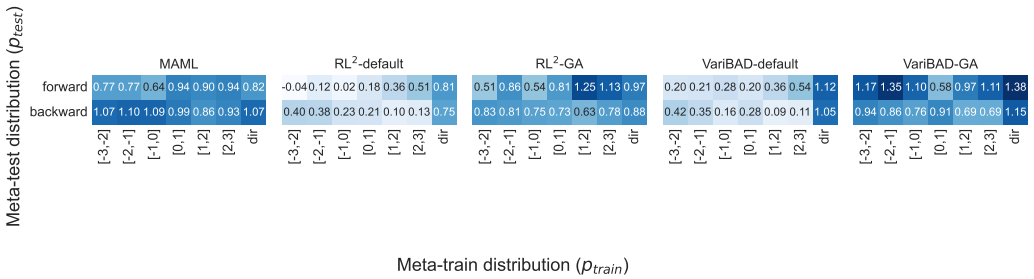

Figure 7: Consistency scores of different meta-RL algorithms on cheetah direction; single-task setting.

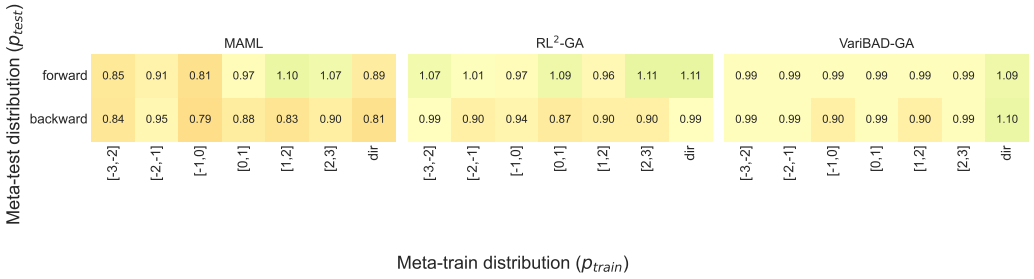

Figure 8: Consistency rates of different meta-RL algorithms on cheetah direction; single-task setting.

C In-distribution Adaptation of Context-based Methods

We compare how context-based methods adapt to in-distribution test tasks via its default setting or gradient adaptation. For example, in-distribution adaptation curves of VariBAD on cheetah-velocity and sparse-navigation are shown in Figures 21 and 22 respectively. On some tasks (especially sparse-navigation tasks), We can observe a dip in performance of VariBAD-GA, which indicates that it's less stable compared to the default adaptation setting without gradient update.

D Behavior Analysis on Sparse Navigation

As shown in Section 4.2 and 4.3, none of the meta-RL algorithms tested can consistently solve the navigation task with sparse reward under the OOD setting, but the context-based methods with gradient adaptation perform better than MAML. We thus look deeper into the behaviors of different methods during OOD adaptation to investigate what's the reason of this phenomenon.

For MAML, we found that the gradients of all network parameters remain 0 during the whole adaptation process, thus it fails to learn anything during OOD adaptation. This is caused by MAML only exploring a very restricted area in the direction of the meta-training goal space, thus not collecting any trajectories with non-zero returns in the direction of the OOD goal space for policy update.

On the contrary, we found that context-based methods explore better than MAML. Figure 23 and 24 show VariBAD's behaviors with its default adaptation setting and via gradient adaptation respectively, where a model is meta-trained to go left and adapted to go right. Under its default adaptation setting, VariBAD follows a fixed path to the left, and fails to explore other areas. However, as gradient update begins, VariBAD starts to randomly explore the space, and sometimes ends up in the right direction. Although VariBAD also just gets trajectories with zero returns before it finds the right direction during gradient adaptation, there are some components inside it that implicitly encourage exploration via parameter update: (1) the VAE is updated to fit to the new transition distribution, which can help the policy explore by providing new context input; (2) the critic module in the policy network is

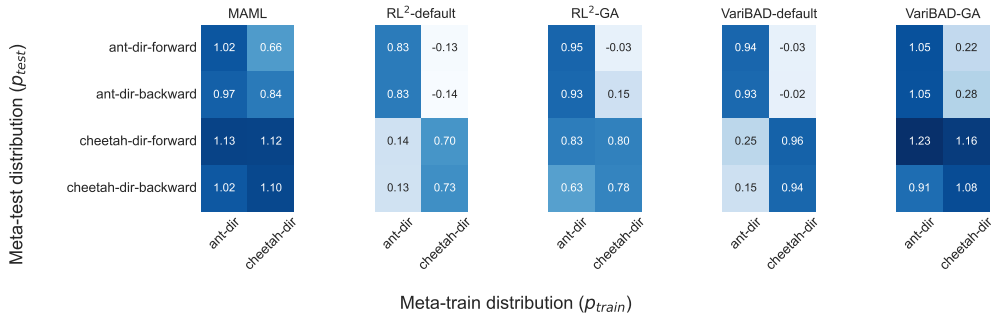

Figure 9: Consistency scores of different meta-RL algorithms on cheetah direction ↔ ant direction; single-task setting.

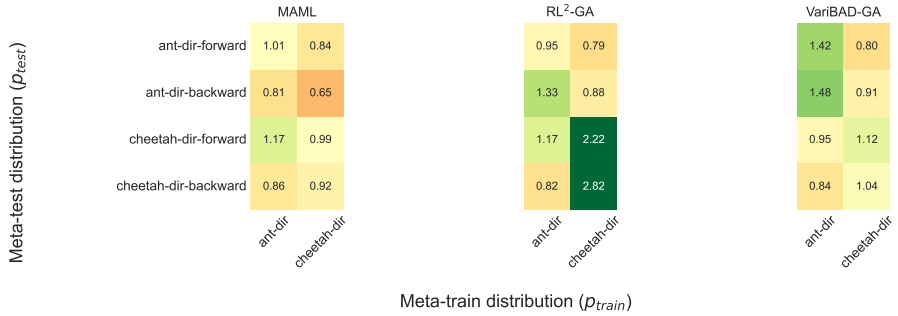

Figure 10: Consistency rates of different meta-RL algorithms on cheetah-direction ↔ ant-direction; single-task setting.

updated for better state value estimation, which encourages parameter update in the policy to try new behaviors. These components which implicitly encourages exploration are missing in MAML, which may explain why context-based methods can get much better performance than MAML on the sparse navigation task. However, as the effect of parameter update on the exploration behavior is quite random, we can not guarantee any convergence to the right goal direction, which may explain why they still can not stably match the final performance of in-distribution learning.

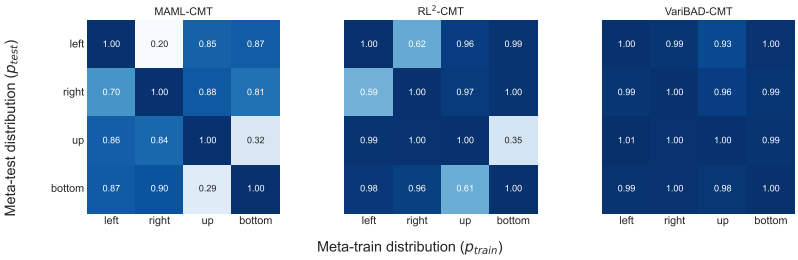

Figure 11: Consistency scores of different meta-RL algorithms on 2D navigation; multi-task setting.

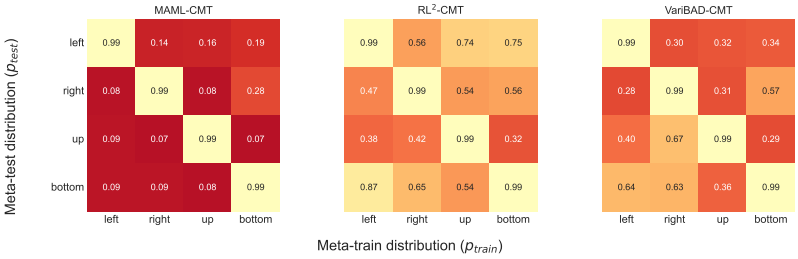

Figure 12: Consistency rates of different meta-RL algorithms on 2D navigation; multi-task setting.

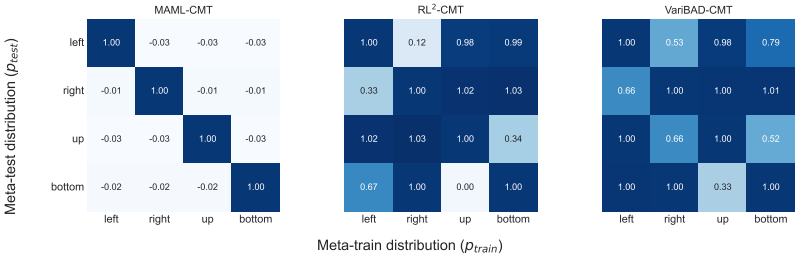

Figure 13: Consistency scores of different meta-RL algorithms on 2D sparse navigation; multi-task setting.

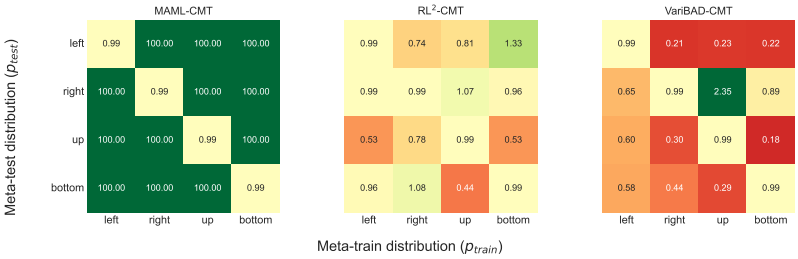

Figure 14: Consistency rates of different meta-RL algorithms on 2D sparse navigation; multi-task setting.

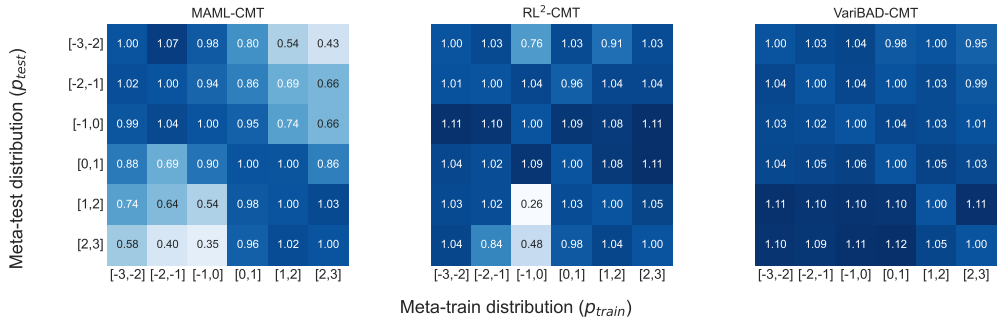

Figure 15: Consistency scores of different meta-RL algorithms on cheetah velocity; multi-task setting.

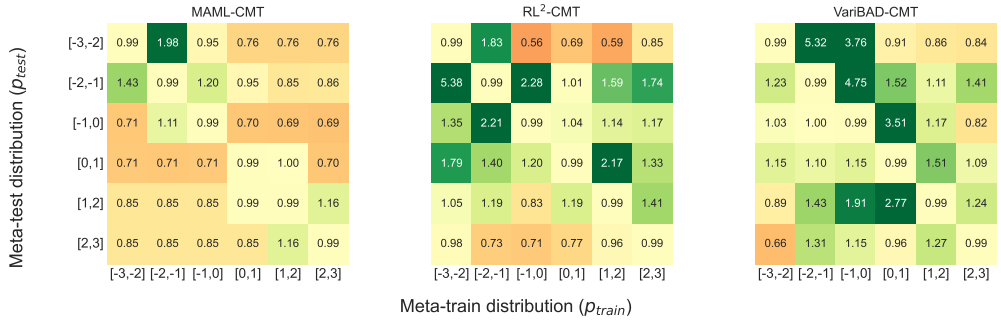

Figure 16: Consistency rates of different meta-RL algorithms on cheetah velocity; multi-task setting.

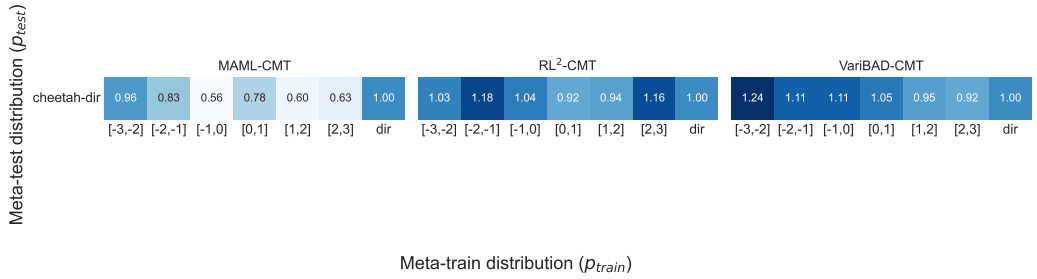

Figure 17: Consistency scores of different meta-RL algorithms on cheetah direction; multi-task setting.

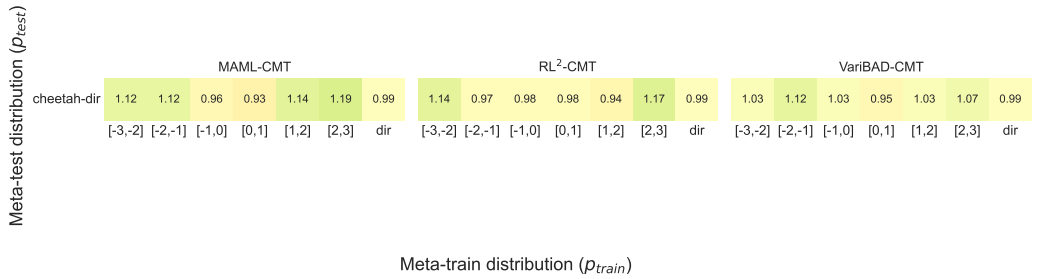

Figure 18: Consistency rates of different meta-RL algorithms on cheetah direction; multi-task setting.

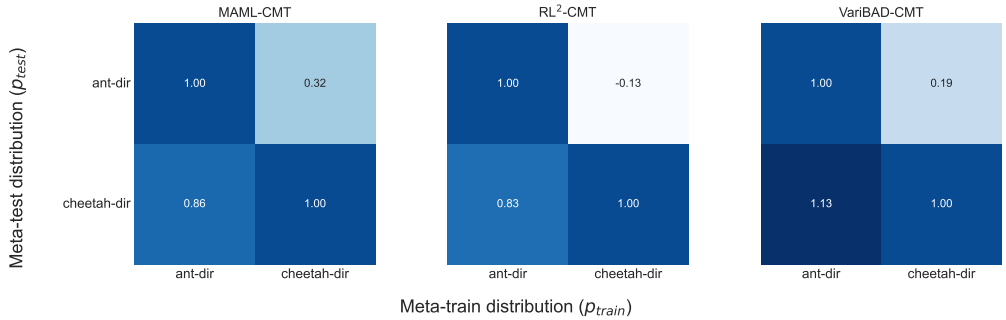

Figure 19: Consistency scores of different meta-RL algorithms on cheetah-direction $\leftrightarrow$ ant-direction; multi-task setting.

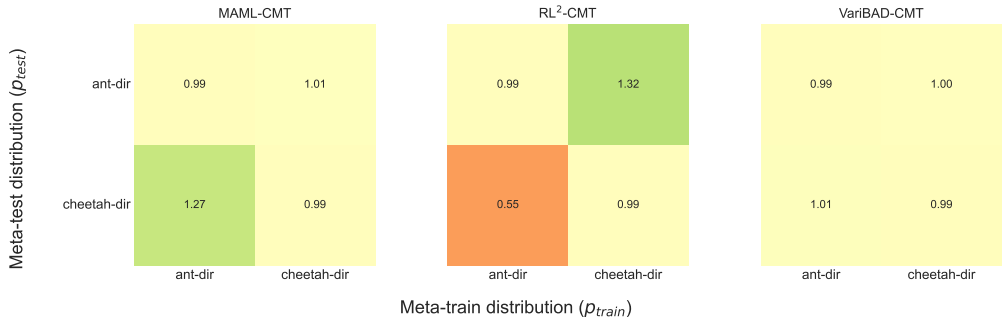

Figure 20: Consistency rates of different meta-RL algorithms on cheetah-direction $\leftrightarrow$ ant-direction; multi-task setting.

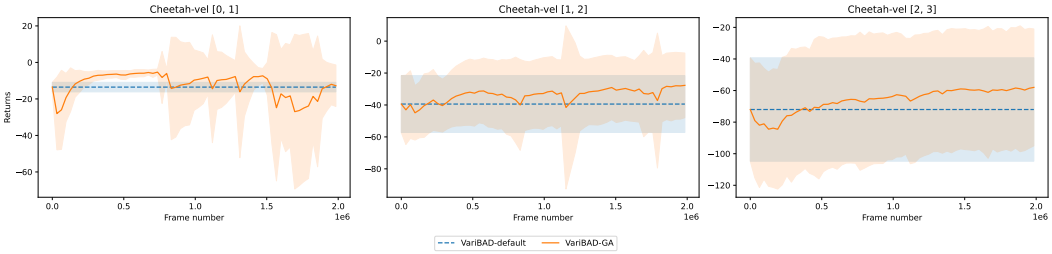

Figure 21: In-distribution adaptation curves of VariBAD on cheetah-velocity $[0, 1]$, $[1, 2]$, $[2, 3]$. The blue dashed lines represent VariBAD's default adaptation performance. These lines are flat because context-based methods by default have no gradient update during adaptation. The orange lines represent how VariBAD's adaptation performance changes via gradient update.

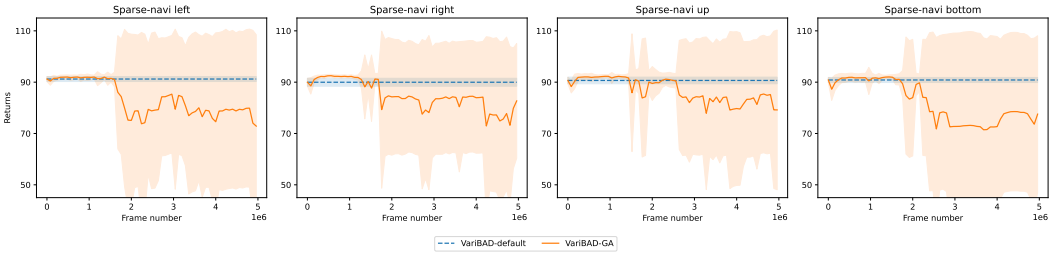

Figure 22: In-distribution adaptation curves of VariBAD on sparse-navigation to different directions.

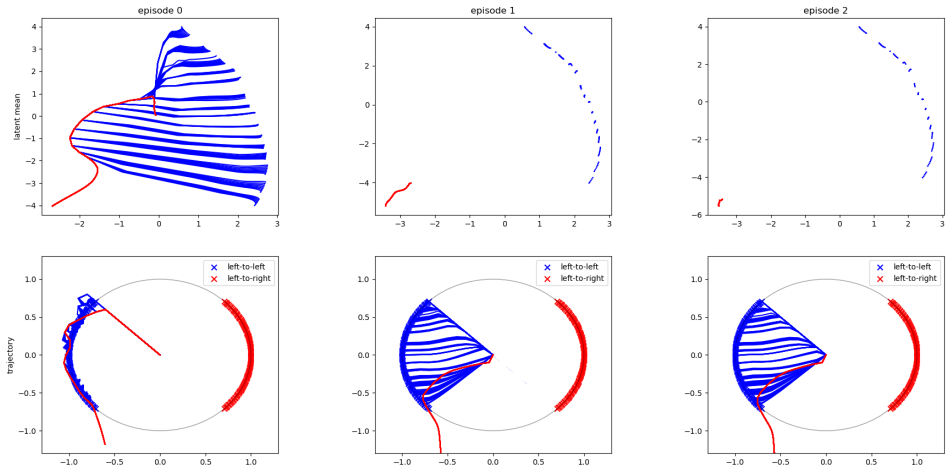

Figure 23: Visualization of VariBAD's behaviors on sparse navigation under its default adaptation setting. Each column shows the results of one episode within an RNN rollout. The first row shows how the inferred context with 2 dimensions evolves (starting from $[0, 0]$ in the first episode) within each episode. The second row shows the trajectory (starting from the origin) of the agent in each episode. The model is meta-trained on sparse navigation to left, and we uniformly sample 100 goals in both left and right for test.

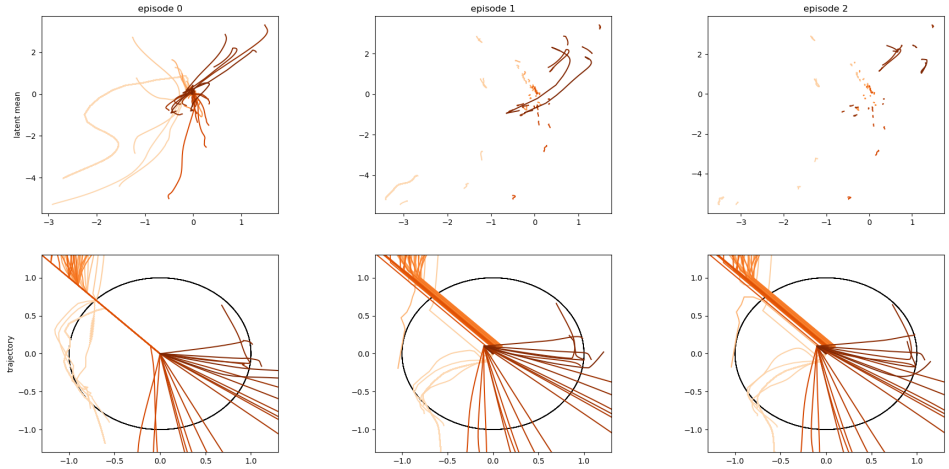

Figure 24: Visualization of VariBAD's behaviors on sparse navigation via gradient adaptation. Each column shows the results of one episode within an RNN rollout. The first row shows how the inferred context with 2 dimensions evolves within each episode. The second row shows the trajectory of the agent in each episode. The model is meta-trained on sparse navigation to left and adapted to right. The lighter color corresponds to the earlier stage during adaptation.