RL^3: Boosting Meta Reinforcement Learning via RL inside RL^2

Abhinav Bhatia, Samer B. Nashed, Shlomo Zilberstein

Keywords: Meta-reinforcement learning

Summary

Meta reinforcement learning (Meta-RL) methods such as RL^2 have emerged as promising approaches for learning data-efficient RL algorithms tailored to a given task distribution. However, they show poor asymptotic performance and struggle with out-of-distribution tasks because they rely on sequence models, such as recurrent neural networks or transformers, to process experiences rather than summarize them using general-purpose RL components such as value functions. In contrast, traditional RL algorithms are data-inefficient as they do not use domain knowledge, but do converge to an optimal policy in the limit. We investigate the hypothesis that incorporating action-values, learned per task via traditional RL, in the inputs to Meta-RL systems that solve the meta-level decision process via an 'outer-loop' deep RL algorithm and a sequence model (e.g. recurrent network, transformer) has a positive effect on the above shortcomings. Using an example implementation, called RL^3, we demonstrate that this strategy earns greater cumulative reward in the long term compared to RL^2 while drastically reducing meta-training time and generalizing better to out-of-distribution tasks. Experiments are conducted on both custom and benchmark discrete domains from the Meta-RL literature that exhibit a range of short-term, long-term, and complex dependencies.

Contribution(s)

1. A thorough investigation of the hypothesis that augmenting Meta-RL inputs with task specific Q-value estimates improves performance across several metrics, which to some may be surprising as this information is already latent within the original input sequence.
   Context: Although some results we present are strong, *this paper is not attempting to present a state-of-the-art Meta-RL system across the board*. We are interested in determining the effects of augmenting the typical Meta-RL inputs of sequence models solving the 'outer-loop' using deep RL algorithms with task specific Q-value estimates, without using other privileged information or extra resources which in practice often increase performance significantly and are in theory *compatible* with this work.
2. This paper presents thorough theoretical, empirical, and logical arguments for the effectiveness of Q-estimate state-augmentation; significant attention is devoted to understanding, from different perspectives, why this method achieves the results we see empirically.
   Context: Previous papers have at times highlighted the importance of object-level value estimation for successful Meta-RL, though never in the form of an algorithm such as RL^3. This paper presents a unique method, set of experiments, and additional analysis complementing existing research in our effort to better understand the capabilities and properties of Meta-RL systems that learn to solve the meta-level decision process via deep RL algorithms with sequence models (e.g. recurrent networks, transformers).

RL^3: Boosting Meta Reinforcement Learning via RL inside RL^2

Abhinav Bhatia[1], **Samer B. Nashed**[1,2,3], **Shlomo Zilberstein**[1]
abhinavbhati@cs.umass.edu, samer.nashed@mila.quebec, shlomo@cs.umass.edu

[1]**Manning College of Information and Computer Sciences, University of Massachusetts Amherst**
[2]**Université de Montréal**
[3]**Mila - Quebec Artificial Intelligence Institute**

Abstract

Meta reinforcement learning (Meta-RL) methods such as RL^2 have emerged as promising approaches for learning data-efficient RL algorithms tailored to a given task distribution. However, they show poor asymptotic performance and struggle with out-of-distribution tasks because they rely on sequence models, such as recurrent neural networks or transformers, to process experiences rather than summarize them using general-purpose RL components such as value functions. In contrast, traditional RL algorithms are data-inefficient as they do not use domain knowledge, but do converge to an optimal policy in the limit. We propose RL^3, a principled hybrid approach that incorporates action-values, learned per task via traditional RL, in the inputs to Meta-RL. We show that RL^3 earns a greater cumulative reward in the long term compared to RL^2 while drastically reducing meta-training time and generalizes better to out-of-distribution tasks. Experiments are conducted on Meta-RL benchmarks and custom discrete domains that exhibit a range of short-term, long-term, and complex dependencies.

1 Introduction

Reinforcement learning (RL) has been shown to produce effective policies in a variety of applications, including both virtual (Mnih et al., 2015) and embodied (Schulman et al., 2017; Haarnoja et al., 2018) systems. However, traditional RL algorithms have three major drawbacks: they can be slow to converge, require a large amount of data, and often have difficulty generalizing to out-of-distribution (OOD) tasks not practiced during training. These shortcomings are especially glaring in settings where the goal is to learn policies for a collection or distribution of problems that share some similarities, and for which traditional RL must start from scratch for each problem. For example, many robotic manipulation tasks require interacting with an array of objects with similar but not identical shapes, sizes, weights, materials, and appearances, such as mugs and cups. It is likely that effective manipulation strategies for these tasks will be similar, but they may also differ in ways that make it challenging to learn a single policy that is highly successful on all instances. Recently, meta-reinforcement learning—often dubbed "learning to learn"— has been proposed as an approach to mitigate these shortcomings by deriving RL algorithms (or Meta-RL policies) that *adapt* efficiently to a distribution of tasks that share some common structure (Duan et al., 2016; Wang et al., 2016).

While Meta-RL systems represent a significant improvement over traditional RL in such settings, they still require large amounts of data during meta-training time, can have poor asymptotic performance during adaptation, and may generalize poorly to tasks not represented in the meta-training distribution. This is partly because they rely on black-box sequence models like recurrent neural

networks or transformers to process experience data. These models cannot handle arbitrary amounts of data effectively and lack integrated general-purpose RL components that could induce a broader generalization bias.

Hence, we propose RL^3, an approach that embeds the strengths of traditional RL within Meta-RL. Table 1 highlights our primary aims and the foremost insight informing our approach. The key idea in RL^3 is an additional 'object-level' RL procedure executed within the Meta-RL architecture that computes task-specific optimal Q-value estimates as supplementary inputs to the meta-learner, in conjunction with sequences of states, actions and rewards. In principle, our approach allows the meta-learner to learn how to optimally fuse raw experience data with summarizations provided by the Q-estimates. Ultimately, RL^3 leverages Q-estimates' generality, ability to compress large amounts of experiences into useful summaries, direct actionability, and asymptotic optimality to enhance long-term performance and OOD generalization and drastically reduce meta-training time.

While Q-value estimates can be injected into any other Meta-RL algorithm, for clarity of exposition, we implement RL^3 by injecting Q-value estimates into one of the most popular and easily understood Meta-RL algorithm, RL^2 (Duan et al., 2016), hence, the name RL^3. However, it should be noted that *our baseline implementation of RL^2 includes significant enhancements* like using transformers instead of LSTMs to improve long-context reasoning, in addition to incorporating numerous recommendations from Ni et al. (2022) that have been shown to make even recurrent model-free RL algorithms like RL^2 competitive with strong Meta-RL baselines like VariBAD (Zintgraf et al., 2020).

The primary contribution of this paper is a proof-of-concept that injecting Q-estimates obtained via traditional object-level RL, alongside experience histories, leads to improved performance, which to some may be surprising as this information is already latent within the original input sequence. Specifically, we observe higher long-term returns and better OOD generalization, while maintaining short-term efficiency in Meta-RL agents like RL^2 that run an 'outer-loop' deep RL algorithm and use sequence models like LSTMs or transformers. We further demonstrate that our approach can also work with an abstract, or coarse, representation of the object-level MDP. We experiment with discrete domains that both reflect the challenges faced by Meta-RL and simultaneously allow transparent analysis of the results. Finally, we examine the key insights that inform our approach and show theoretically that object-level Q-values are directly related to the optimal meta-value function.

2 Related Work

Although Meta-RL is a fairly new topic of research, the general concept of meta-learning is decades old (Vilalta & Drissi, 2002), which, coupled with a significant number of design decisions for Meta-RL systems, has created a large number of different proposals for how systems ought to best exploit the resources available within their deployment contexts (Beck et al., 2023). At a high level, most Meta-RL algorithms can be categorized as either parameterized policy gradient (PPG) models (Finn et al., 2017; Li et al., 2017; Sung et al., 2017; Al-Shedivat et al., 2018; Gupta et al., 2018; Yoon et al.,

Table 1: RL^3 combines the strengths of RL^2 and traditional RL. Like RL^2, RL^3 uses finite-context sequence models to represent data-efficient RL algorithms, optimized for tasks within a specified distribution. However, RL^3 also includes a general-purpose RL routine that distills arbitrary amounts of data into optimal value-function estimates during adaptation. This improves long-term reasoning and OOD generalization.

	RL	**RL^2**	**RL^3**
Short-Term Efficiency	✗	✓	✓
Long-Term Performance	✓	✗	✓
OOD Generalization	✓	✗	✓
	(General Purpose)		(Improved)

2018; Stadie et al., 2018; Vuorio et al., 2019; Zintgraf et al., 2019; Raghu et al., 2019; Kaushik et al., 2020; Ghadirzadeh et al., 2021; Mandi et al., 2022) or black box models (Duan et al., 2016; Heess et al., 2015; Wang et al., 2016; Foerster et al., 2018; Mishra et al., 2018; Humplik et al., 2019; Fakoor et al., 2020; Yan et al., 2020; Zintgraf et al., 2020; Liu et al., 2021; Emukpere et al., 2021; Beck et al., 2022). PPG approaches assume that the underlying learning process is best represented as a policy gradient, where the set of parameters that define the underlying algorithm ultimately form a differentiable set of meta-parameters that the Meta-RL system may learn to adjust. The additional structure provided by this assumption, combined with the generality of policy gradient methods, means that typically PPG methods retain greater generalization capabilities on out-of-distribution tasks. However, due to their inherent data requirements, PPG methods are often slower to adapt and initially train.

In this paper we focus on black box models, which represent the meta-learning function as a neural network, often a recurrent neural network (RNN) (Duan et al., 2016; Heess et al., 2015; Wang et al., 2016; Humplik et al., 2019; Fakoor et al., 2020; Yan et al., 2020; Zintgraf et al., 2020; Liu et al., 2021) or a transformer (Mishra et al., 2018; Wang et al., 2021; Melo, 2022). There are also several hybrid approaches that combine PPG and black box methods, either during meta-training (Ren et al., 2023) or fine-tuning (Lan et al., 2019; Xiong et al., 2021). Using black box models simplifies the process of augmenting meta states with Q-estimates and allows us to retain relatively better data efficiency while relying on the Q-value injections for better long-term performance and generalization.

Meta-RL systems may also leverage extra information available during training, such as task identification (Humplik et al., 2019; Liu et al., 2021). Such 'privileged information' can of course lead to more performant systems, but is not universally available. As our hypothesis does not rely on the availability of such information, we expect our approach to be orthogonal to, and compatible with, such methods. Black box Meta-RL systems that do not use privileged information still vary in several ways, including the choice between on-policy and off-policy learning and, in systems that use neural networks, the choice between transformers (Vaswani et al., 2017) and RNNs (Elman, 1990; Hochreiter & Schmidhuber, 1997; Cho et al., 2014).

The most relevant methods to our work are end-to-end methods, which use a single function approximator to subsume both learner and meta-learner, such as RL^2 (Duan et al., 2016), L2L (Wang et al., 2016), SNAIL (Mishra et al., 2018), and E-RL^2 (Stadie et al., 2018), and methods that exploit the formal description of the Meta-RL problem as a POMDP or a Bayes-adaptive MDP (BAMDP) (Duff, 2002). These methods attempt to learn policies conditioned on the BAMDP belief state while also approximating this belief state by, for example, variational inference (VariBAD) (Zintgraf et al., 2020; Dorfman et al., 2020), or random network distillation on belief states (HyperX) (Zintgraf et al., 2021). Or, they simply encode enough experience history to approximate POMDP beliefs (RL^2) (Duan et al., 2016; Wang et al., 2016).

Our proposed method is an end-to-end system that exploits the BAMDP structure of the Meta-RL problem by spending a small amount of extra computation to provide inputs to the end-to-end learner that more closely resemble important constituents of BAMDP value functions. Thus, the primary difference between this work and previous work is the injection of Q-value estimates into the Meta-RL agent state at each meta-step, in addition to the state-action-reward histories. In this work, our approach, RL^3, is implemented by simply injecting Q-value estimates into RL^2 alongside experience history, although any other Meta-RL algorithm can be used.

3 Background and Notation

In this section, we briefly cover some notation and concepts upon which this paper is built.

3.1 Partially Observable MDPs

We use the standard notation defining a Markov decision process (MDP) as a tuple $M = \langle S, A, T, R \rangle$, where S is a set of states, A is a set of actions, T is the transition function, and R is the

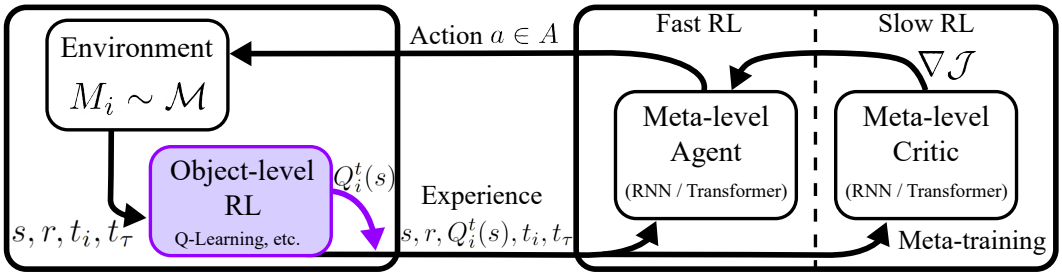


Figure 1: Overview diagram of RL^3. Black entities represent standard components from RL^2, and purple entities represent additions for RL^3. M_i is the current MDP; s is a state; r is a reward; t_i and t_τ are the amount of time spent experiencing the current MDP and current episode, respectively; Q_i^t is the Q-value estimate for MDP i after t actions; $\nabla \mathcal{J}$ is the policy gradient for meta-training.

reward function. A *partially observable Markov decision process* (POMDP) extends MDPs to settings with partially observable states. A POMDP is described as a tuple $\langle S, A, T, R, \Omega, O \rangle$, where S, A, T, and R are as in an MDP. In a POMDP, the agent receives observations $\omega \in \Omega$ instead of direct state information, with an observation function $O(\omega|s', a)$ representing the probability of observing ω after taking action a and transitioning to state s'. POMDPs can also be formulated as continuous-state belief MDPs, where a belief state $b \in \Delta^{|\mathcal{S}|}$ is a probability distribution over all possible states. Beliefs are updated online via Bayesian inference: $b'(s'|b, a, \omega) \propto O(\omega|s', a) \sum_{s \in S} T(s'|s, a)\, b(s)$ as new observations are received. The policy maps beliefs to actions, $\pi : \Delta^{|\mathcal{S}|} \to \mathcal{A}$.

3.2 Reinforcement Learning

Reinforcement learning (RL) agents learn an optimal policy in an MDP with unknown dynamics using only transition and reward feedback. This is often done by incrementally estimating the optimal action-value function $Q^*(s, a)$ (Watkins & Dayan, 1992), which satisfies the Bellman optimality equation $Q^*(s, a) = \mathbb{E}_{s'}[R(s, a) + \gamma \max_{a' \in A} Q^*(s', a')]$. In large or continuous state spaces, it is common to use deep neural networks to represent the action-value function (Mnih et al., 2015). We denote the vector of Q-estimates for all actions at state s as $Q(s)$, and after t feedback steps as $Q^t(s)$. Q-learning is known to converge asymptotically (Sutton & Barto, 2018), provided each state-action pair is explored sufficiently. As a rough statement, $||Q^t(s) - Q^*(s)||_\infty$ is proportional to $\approx \frac{1}{\sqrt{t}}$, with strong results on convergence error available (Szepesvári, 1997; Kearns & Singh, 1998; Even-Dar et al., 2003). Typically, the *formal objective* in RL is to optimize the value of the final policy, i.e., the cumulative reward per episode, disregarding the cumulative reward missed (or *regret*) during learning due to suboptimal actions taken by the *behavior policy* used for exploration.

3.3 Meta Reinforcement Learning

While standard model-free RL seeks a single optimal policy per MDP, Meta-RL aims to train strategies that rapidly adapt across a distribution of related tasks $\mathcal{M}$. This is especially valuable when the agent faces a fixed interaction budget H per task, assuming shared state and action spaces. Formally, Meta-RL optimizes a 'fast' *behavior policy* π, also called the *Meta-RL policy*, to maximize cumulative reward (or equivalently, minimize regret) during the adaptation period:

$$\mathcal{J}(\pi) = \mathbb{E}_{M_i \sim \mathcal{M}} \Big[\sum_{t=0}^{H} \gamma^t \, \mathbb{E}_{(s_t, a_t) \sim \rho_{i,t}^{\pi}} \big[R_i(s_t, a_t) \big] \Big], \tag{1}$$

where $\rho_{i,t}^{\pi}$ denotes the state-action distribution induced by π in M_i at timestep t. Unlike Markovian policies in standard RL, the Meta-RL policy conditions on the *experience history* $\Upsilon = (s_0, a_0, r_0, \ldots, s_t)$, which spans multiple episodes within each adaptation period.

Theoretically, Meta-RL can be framed as solving a meta-level POMDP in which the task identity M_i is the hidden variable. Augmenting the state with a belief over tasks $b(i)$ yields a Bayes-Adaptive MDP (BAMDP) (Duff, 2002; Ghavamzadeh et al., 2015), where the belief state $[s, b]$ captures all necessary information for optimal action selection. In practice, however, many approaches approximate this mapping without explicit belief modeling, instead relying on recurrent or transformer-based policies. A prominent example of this class is RL^2 (Duan et al., 2016; Wang et al., 2016), which we consider representative of Meta-RL methods that solve the meta-level POMDP by using an outer-loop 'slow' deep RL algorithm e.g., PPO (Schulman et al., 2017) to train a sequence model—such as a recurrent network or transformer—to map experience histories to policies or meta-value functions, thereby optimizing $\mathcal{J}(\pi)$. Our approach builds on this framework by augmenting the experience history with task-specific value estimates, as described below.

4 RL^3

To address the limitations of black box Meta-RL methods, we propose RL^3, a principled approach that leverages (1) the inherent generality of action-value estimates, (2) their ability to compress experience histories into useful summaries, (3) their direct actionability & asymptotic optimality, (4) their ability to inform task identification, and (5) their relation to the optimal meta-value function, in order to enhance out-of-distribution (OOD) generalization and performance over extended adaptation periods. The central, novel mechanism in RL^3 is an additional 'object-level' RL procedure executed within the Meta-RL architecture, shown in Fig. 1, that computes task-specific optimal Q-value estimates $Q_i^t(s_t)$—denoting $\hat{Q}^*$ estimates for task i after t transition samples—and state-action counts as supplementary inputs to the Meta-RL policy in conjunction with the sequence of states, actions, and rewards $(s_0, a_0, r_0, ..., s_t)$. The Q-estimates are computed off-policy and may involve experience replay or model estimation & planning for greater data efficiency. The estimates and the counts are reset at the beginning of each meta-episode as a new task M_i is sampled. We now present a series of key insights informing our approach.

First, estimating action-values is a key component in many general-purpose, or **universal**, RL algorithms, and asymptotically, they *fully* inform optimal behavior *irrespective of domain*. Strategies for optimal exploration-exploitation trade-off are domain-dependent and rely on historical data, yet many exploration approaches use estimated Q-values and some notion of counts *alone*, such as epsilon-greedy, Boltzmann exploration, upper confidence bounds (UCB/UCT) (Auer, 2002; Kocsis & Szepesvári, 2006), count-based exploration (Tang et al., 2017), curiosity-based exploration (Burda et al., 2019), and maximum-entropy RL (Haarnoja et al., 2018). This creates a strong empirical case that using Q-value estimates and state-action counts for efficient exploration has inherent generality.

Second, Q-estimates **summarize experience histories** of arbitrary length *and order* in one constant-size vector. This mapping is many-to-one, and any permutation of transitions ($\langle s, a, r, s' \rangle$ tuples) or episodes in a history of experiences yield the same Q-estimates. Although this compression is lossy, it still "remembers" important aspects of the experienced episodes, such as high-return actions and goal positions (see Fig. 2), since Q-estimates persist across episodes. In conjunction, action counts provide an explicit record of exploration coverage. This simplifies the mapping the meta-agent needs to learn, as Q-estimates and counts together represent a smaller and more salient set of inputs compared to all possible histories with the same implication.

Third, Q-estimates are **actionable**. Estimated off-policy, they explicitly represent the optimal exploitation policy for the current task given the data, insofar as the RL module is data-efficient, relieving the Meta-RL agent from performing such calculations inside the transformer or RNN. Over time, Q-estimates become more reliable and directly indicate the optimal policy, whereas processing raw data becomes more challenging. Provided alongside action counts, they also give the agent an implicit mechanism to modulate trust in Q-estimates as evidence accumulates. Fortunately, by incorporating Q-estimates, the Meta-RL agent can eventually ignore the history in the long run (or towards the end of the interaction period) and simply exploit the Q-estimates by selecting actions greedily.

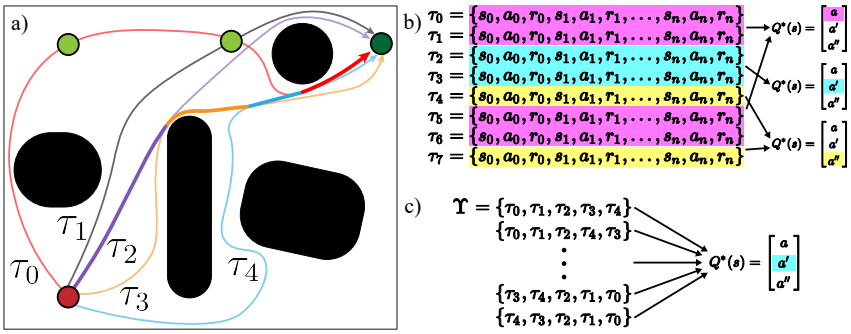


Figure 2: Sub-figure (a) shows a meta-episode in a shortest-path environment where the goal position (green circles) and the obstacles (black regions) may vary across tasks. In this meta-episode, after the Meta-RL agent narrows its belief about the goal position of this task (dark-green circle) having followed a principled exploration strategy (τ_0), it explores potential shorter paths in subsequent episodes ($\tau_1, \tau_2, \tau_3, \tau_4$). Throughout this process, the estimated value-function $\hat{Q}^*$ implicitly "remembers" the goal position and previous paths traversed in a finite-size representation, and updates the shortest path calculation (highlighted in bold) using Bellman backups when paths intersect. Sub-figures (b) and (c) illustrate the many-to-one mapping of object- and meta-level data streams to Q-estimates, and thus their utility as compression and summarization mechanisms for meta-learning.

Fourth, Q-estimates are **excellent task discriminators** and serve as another line of evidence vis-à-vis maintaining belief over tasks. In a simple domain like Bernoulli multi-armed bandits (Duan et al., 2016), Q-estimates and action counts combined are sufficient for Bayes-optimal behavior even without providing raw experience data—a result surprisingly unstated in the literature to the best of our knowledge (see Appendix A.1). However, Q-estimates and counts may not always be sufficient for Bayes-optimal beliefs (see Gaussian multi-armed bandits example in Appendix A.2). In more complex domains, it is hard to prove the sufficiency of Q-estimates regarding task discrimination. However, via empirical analysis in Appendix D, we argue that i) it is highly improbable for two tasks to have similar Q^* functions and ii) Q-estimates tend to become accurate task predictors in just a few steps. This implies that the meta-agent may use this finite summary for task inference rather than relying completely on arbitrarily long histories, potentially contributing to enhanced performance over long adaptation periods.

It can be theoretically argued that since the meta-agent is a BAMDP *policy*, it is meta-trained to select greedy actions with respect to the BAMDP meta-value function and thus should not require constructing a task-specific plan internally. However, the optimality of the meta action-value function depends on implicitly (or explicitly in some approaches (Humplik et al., 2019; Zintgraf et al., 2020; Dorfman et al., 2020; Zintgraf et al., 2021)) maintaining a Bayes-optimal belief over tasks in the transformer or RNN architecture. This may be challenging if the task distribution is too broad and the function approximator is not powerful enough to integrate experience histories into Bayes-optimal beliefs, or altogether impossible if there is a distribution shift at meta-test time. This latter condition is common in practice and is a frequent target use case for Meta-RL systems. Incorporating task-specific Q-estimates gives the agent a simple alternative (even if not Bayes-optimal) line of reasoning to translate experiences into actions. Incorporating Q-estimates thus **reduces susceptibility to distribution shifts**, since the arguments presented in this section are domain independent.

Finally, Q-estimates often converge far more quickly than the theoretical rate of $1/\sqrt{t}$, allowing them to be useful in the short and medium term, since i) most real-world domains contain significant determinism, ii) it is not necessary to estimate Q-values for states unreachable by the optimal policy, and iii) optimal Meta-RL policies may represent active exploration strategies in which Q-estimates converge faster, or evolve in a manner leading to quicker task identification. This is intuitively apparent in shortest-path problems, as illustrated in Fig. 2(a). In a deep neural network, it is difficult to know exactly how Q-estimates will combine with state-action-reward histories when approximating

the meta-value function. However, as we show below, we can write an equation for the meta-value function in terms of these constituent streams of information, which may explain why this function is seemingly relatively easy to learn compared to predicting meta-values from histories alone.

4.1 Theoretical Justification

Throughout this section, without loss of generality, we assume: i) an infinite-horizon setting, ii) that $\mathcal{M}$ denotes the support set of possible MDPs over which a distribution is defined, iii) all MDPs share the same state and action spaces, and iv) rewards are uniformly bounded by $|R_i(s,a)| \leq R_{\max}$. We consider the interpretation of Meta-RL as performing RL on a partially observable Markov decision process (POMDP) in which the partially observable factor is the identity of the object-level MDP. We denote meta-level entities (i.e., those belonging to the POMDP) with an overbar; for example, the meta-level value function $\bar{V}$ and the meta-level belief $\bar{b}$. First, we establish that the optimal meta-level value function is upper bounded by the object-level Q-values.

Proof. For a given state s, there exists a maximum object-level optimal value function $V^*_{\max}(s)$ corresponding to some MDP $M_{\max} \in \mathcal{M}$, such that for all MDPs $M_i \in \mathcal{M}$, $V^*_{\max}(s) \geq V^*_i(s)$. The expected cumulative discounted reward experienced by the agent cannot exceed this most optimistic value, since $\bar{V}^*(\bar{b})$ is a weighted average of the individual value functions $V^{\pi_\theta}(s)$, each of which is upper bounded by $V^*_{\max}(s)$. Thus,

$$\max_{M_i \in \mathcal{M}} V^*_i(s) \;\geq\; \bar{V}^*(\bar{b}) \quad \forall s \in S. \tag{2}$$

Combining the asymptotic accuracy of Q-estimates with Equation 2 yields

$$\lim_{t\to\infty} \max_{a\in A, M_i \in \mathcal{M}} Q^t_i(s,a) \geq \bar{V}^*(\bar{b}) \quad \forall s \in S. \quad \square \tag{3}$$

Furthermore, if the meta-level observation $\bar{\omega}$ includes the Q-value estimates of the current task M_i, it can be shown that as $t \to \infty$, the optimal meta-value function approaches the optimal value function for the current task. Specifically, for any $\epsilon > 0$, there exists $\kappa \in \mathbb{N}$ such that for all $t \geq \kappa$,

$$\left| \max_{a\in A} \left[Q^t_i(s,a) \right] - \bar{V}^*(\bar{b}) \right| \leq \epsilon \quad \forall s \in S. \tag{4}$$

Equation 4 (see proof in Appendix A.3) shows that for $t \geq \kappa$, acting greedily with respect to Q^*_i leads to Bayes-optimal behavior, and explicitly maintaining the Bayes-optimal belief over tasks is no longer required. This implies that the experience history Υ can be ignored at that point.

Moreover, Equation 4 implies that for $t < \kappa$,

$$\bar{V}^*(\bar{b}) = \max_{a\in A} \left[Q^t_i(s,a) \right] + \varepsilon_i(\Upsilon) \tag{5}$$

where $\varepsilon_i(\Upsilon)$ denotes the error in Q-value estimates as a function of the experience history Υ. While this error diminishes as $t \to \infty$, in the short run, a function $f(\Upsilon)$ could be learned to either estimate the error or directly approximate $\bar{V}^*(\bar{b})$. The better performance of RL^3 could be explained by either the error $\varepsilon_i(\Upsilon)$ being simpler to estimate, or the meta-agent's behavior being more robust to errors in $\varepsilon_i(\Upsilon)$ when Q-estimates are supplied directly as inputs, compared to errors arising from approximating $\bar{V}^*(\bar{b})$ without such inputs. Additionally, this approach benefits from the fact that the convergence rate of Q-estimates suggests a natural, predictable schedule of shifting reliance from $f(\Upsilon)$ to $Q^t_i(s)$ as $t \to \infty$. However, we do not impose this structure explicitly in the network and instead allow it to learn implicitly how much to rely on the Q-estimates, informed by the context and supported by the provided action counts as supplementary signals.

Finally, we note that achieving accurate approximation of $\bar{V}^*(\bar{b})$ as $t \to \infty$ reduces the error in meta-value estimation for all preceding belief states, since meta-values for consecutive beliefs $\bar{b}$ and $\bar{b}'$ are linked via the Bellman equation for BAMDPs (see Appendix A.3):

$$\bar{V}^*(\bar{b}) = \max_{a\in A} \Big[\sum_{M_i \in \mathcal{M}} \bar{b}(i) R_i(s,a) + \gamma \sum_{\bar{\omega}\in\bar{\Omega}} \Pr(\bar{\omega}|\bar{b},a) \bar{V}^*(\bar{b}') \Big], \tag{6}$$

This dependency helps meta-training in RL^3 when using temporal-difference learning algorithms. Without conditioning on Q-estimates, the error in $\bar{V}^*(\bar{b})$ could instead grow as $t \to \infty$, since the meta-critic must condition on an increasingly large experience history Υ. This could destabilize meta-value learning across all preceding belief states during meta-training.

4.2 Implementation

Implementing RL^3 involves replacing each MDP in the task distribution with a value-augmented MDP (VAMDP) and solving the resulting distribution using RL^2. Each VAMDP has the same action space and reward function as the original MDP. The value-augmented state $\hat{s}_t \in S \times \mathbb{R}^k \times \mathbb{I}^k$ comprises the object-level state s_t, k real values and k integer values for the Q-estimates $Q^t(s_t, a)$ and action counts $N^t(s_t, a)$ for each action. In practice, we represent Q-estimates in a numerically stabilized form by decomposing them into their maximum value and centered differences, i.e., providing $\max_a Q^t(s_t, a)$ and the offsets $Q^t(s_t, a) - \max_a Q^t(s_t, a)$, which correspond to the value function and advantages, respectively. This reparametrization is numerically equivalent to raw Q-values but improves training stability by normalizing input scale. When S is discrete, s_t is encoded as an $|S|$-dimensional one-hot vector. The value-augmented state space is thus continuous. In the VAMDP transition function, the object-level state s evolves according to the original dynamics, while Q-estimate dynamics depend on T, R, and the specific RL algorithm used. A VAMDP episode spans the full interaction period with the MDP, as Q-estimates evolve across multiple episodes. In code, a VAMDP environment is implemented as a wrapper over the base MDP. Pseudocode, additional details, and hyperparameters for RL^2 and RL^3 are provided in Appendix B.

5 Experiments

We compare RL^3 to our enhanced implementation of RL^2. In our implementation, we replace LSTMs with transformers in both the meta-actor and meta-critic for the purpose of mapping experiences to actions and meta-values, respectively. This is done to improve RL^2 's ability to handle long-term dependencies instead of suffering from vanishing gradients. Moreover, RL^2-transformer trains significantly faster than RL^2-LSTM. Second, we include in the state space the total number of interaction steps and the total number of steps within each episode during a meta-episode (see Fig. 1). Third, we use PPO (Schulman et al., 2017) for training the meta actor-critic, instead of TRPO (Schulman et al., 2015). These modifications and other minor-implementation details incorporate the recommendations made by Ni et al. (2022), who show that model-free recurrent RL is competitive with strong Meta-RL approaches such as VariBAD (Zintgraf et al., 2020), if implemented properly. RL^3 simply applies the modified version of RL^2 to the distribution of value-augmented MDPs explained in section 4.2. Within each VAMDP, our choice of object-level RL is a model-based algorithm to maximize data efficiency – we estimate a tabular model of the environment and run finite-horizon value-iteration using the model. Once again, we emphasize that the core of our approach, which is augmenting MDP states with action-value estimates, is not inherently tied to RL^2 and is orthogonal to most other Meta-RL research. VAMDPs can be plugged into any base Meta-RL algorithm with a reasonable expectation of improving it.

In our test domains, each meta-episode involves procedurally generating an MDP according to a parameterized distribution, which the meta-actor interacts with for a fixed adaptation period, or interaction budget, H. This interaction might consist of multiple object-level episodes of variable length, each of which are no longer than a maximum task horizon. For a given experiment, each approach is trained on the same series of MDPs. Each experiment is done for 3 seeds and the results of the median performing model are reported. For testing, each approach is evaluated on an identical set of 1000 MDPs distinct from the training MDPs. For testing OOD generalization, MDPs are generated from distributions with different parameters than in training. We select three discrete domains for our experiments, which cover a range of short-term, long-term, and complex dependencies. These domains both reflect the challenges faced by Meta-RL and simultaneously allow transparent analysis of the results.

Table 2: Test scores (mean $\pm$ standard error) for Bandits domain and the †OOD variation.

Budget H	**RL^2**	**RL^3**	**RL^3 (Markov)**
100	76.9 ± 0.6	77.5 ± 0.5	75.2 ± 0.5
500	392.1 ± 2.5	393.2 ± 2.7	391.75 ± 2.6
500†	430.2 ± 2.8	$\mathbf{434.9} \pm 2.8$	433.7 ± 2.8

Bernoulli Bandits: We use the same setup described by Duan et al. (2016) with $k = 5$ arms. To test OOD generalization, we generate bandit tasks by sampling success probabilities from $\mathcal{N}(0.5, 0.5)$. We should note that this is an easy domain and serves as a sanity check to ensure that Q-value estimates do not hurt RL^3, causing inferior performance.

Random MDPs: We use the same setup described by Duan et al. (2016). The MDPs have 10 states, 5 actions, and task horizon 10. The rewards and transition probabilities are drawn from a normal and a flat Dirichlet distribution ($\alpha = 1.0$), respectively. OOD test MDPs use Dirichlet $\alpha = 0.25$. We should note that this domain is particularly challenging for RL^3 due to the high degree of stochasticity and thus the slower convergence rate of Q-estimates.

GridWorld Navigation: A set of navigation tasks in a 2D grid environment. We experiment with 11x11 (121 states) and 13x13 (169 states) grids. The agent starts in the center and needs to navigate through obstacles to a single goal. The grid also contains slippery tiles, dangerous tiles and warning tiles. See Fig. 4(a) for an example of a 13x13 grid. The state representation is coordinates (x, y). To test OOD generalization, we vary parameters including the stochasticity of actions, density of obstacles and the number of dangerous tiles. For this domain, we consider an additional variation of RL^3, called RL^3-*coarse* where a given grid is partitioned into clusters of 2 adjacent tiles (or abstract states), which are used *solely* for the purpose of estimating the object-level Q-values. Our goal is to test whether coarse-level Q-value estimates are still useful to the Meta-RL policy. The domains and the abstraction strategy are described in greater detail in Appendices E and B.3, respectively.

6 Results

In summary, we observe that beyond matching or exceeding the performance of RL^2-transformer in all test domains i) RL^3 shows better OOD generalization, which we attribute to the increased generality of the Q-value representation, ii) the advantages of RL^3 increase with longer interactions periods and less stochastic tasks, which we attribute to the increased accuracy of the Q-value estimates, iii) RL^3 performs well even with coarse-grained object-level RL over abstract states with substantial computational savings, showing minimal drop in performance in most cases, and iv) RL^3 shows faster meta-training.

Bandits: Fig 2 shows the results for this sanity-check domain. For $H = 100$ and $H = 500$, both approaches perform comparably. However, the OOD generalization for RL^3 is slightly better. We also experiment with a Markovian version of RL^3, where a feed-forward neural network is conditioned only on the Q-estimates and action-counts, since those are sufficient for Bayes-optimal behavior in this domain. As expected, the results are similar to regular RL^3 .

MDPs: Figures 3(a) and 3(b) show the results for the MDPs domain. In Figure 3(a), we see that for relatively short budgets, $H \leq 500$, both RL^3 and RL^2-transformer perform comparably on in-distribution problems, with RL^3 performing slightly better on OOD tasks. We suspect that, due to the short budgets and highly stochastic domain, Q-estimates do not converge enough to be very useful for RL^3 . However, as the budget increases, we see that RL^3 continues to improve while RL^2-transformer actually becomes worse and the performance gap on both in-distribution and OOD tasks becomes significant. Overall, we see that RL^3 *preserves asymptotic scaling properties of traditional RL while simultaneously maintaining strong OOD performance*. Moreover, RL^3 it is able to learn meta-policies much more efficiently. Figure 3(b) shows the number of iterations of PPO RL^3 takes

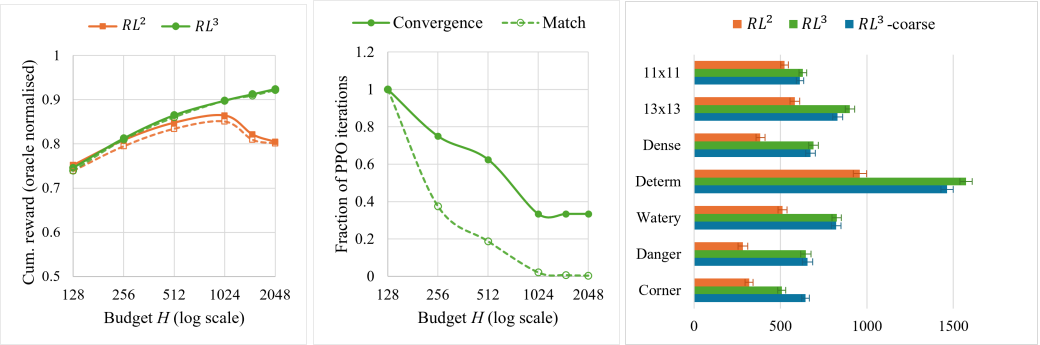


Figure 3: Results for the MDPs and GridWorlds domains. Figure 3a shows the average cumulative reward (negligible standard error) earned as a fraction of the oracle policy for in-distribution (solid) and OOD (dashed) tasks; Figure 3b shows the fraction of RL^2-transformer meta-training iterations that RL^3 requires (variance is insignificant across seeds) to match RL^2-transformer performance or fully converge, both as functions of the adaptation period H. Note the log horizontal axis on both plots. Figure 3c shows the average cumulative reward ($\pm$ standard error) earned by RL^2-transformer, RL^3 and RL^3-coarse agents on several variations of the GridWorlds domain.

to converge completely, as well as to match the performance of RL^2-transformer, measured as a fraction of the time it takes for RL^2-transformer to converge. This advantage of RL^3 is again most pronounced for longer adaptation periods, but we still do observe significant meta-training speedup on even moderate ones. The training curves are available in Appendix C. Overall, it is clear that as adaptation periods grow, RL^3 achieves nearer-to-optimal policies in a fraction of the meta-training time and maintains better OOD generalization.

GridWorlds: Fig 3(c) shows the results for the GridWorlds domain. On 11x11 grids with $H = 250$, RL^3 significantly outperforms RL^2-transformer. On 13x13 grids with $H = 350$, the performance margin is even greater, showing that while RL^2-transformer struggles with a greater number of states, a longer adaptation period and more long-term dependencies, RL^3 can take advantage of the Q-estimates to overcome the challenge. We also test the OOD generalization of both approaches in different ways by varying certain parameters of the 13x13 grids, namely, increasing the obstacle density (DENSE), making actions on non-water tiles deterministic (DETERMINISTIC), increasing the number of wet 'W' tiles (WATERY), increasing the number of danger 'X' tiles (DANGEROUS) and having the goal only in the corners (CORNER). On all variations, RL^3 continues to significantly outperform RL^2-transformer. In a particularly interesting outcome, both approaches show improved performance on the DETERMINISTIC variation. However, RL^3 gains 80% more points than RL^2-transformer, which is likely because Q-estimates converge faster on this less stochastic MDP. Conversely, in the WATERY variation, which is more stochastic, both RL^2-transformer and RL^3 lose roughly equal number of points. In each case, RL^3-coarse significantly outperforms RL^2-transformer. In fact, it performs on par with RL^3, even outperforming it on CORNER variation, except on the canonical 13x13 case and its DETERMINISTIC variation, where it scores about 90% of the scores for RL^3. Finally, we see similar meta-training speedups where RL^3 requires just 50% and 30% of the total iterations to match the performance of RL^2-transformer on the 11x11 and 13x13 grids, respectively. The training curves are available in Appendix C.

Fig. 4 shows a sequence of snapshots of a meta-episode where the trained RL^3 agent is interacting with an instance of a 13x13 grid. The first snapshot shows the agent just before reaching the goal for the first time. Prior to the first snapshot, the agent had explored many locations in the grid. The second snapshot shows the next episode just after the agent finds the goal, resulting in value estimates being updated using object-level RL for all visited states. Snapshot 3 shows the agent consequently using the Q-estimates to navigate to the goal presumably by choosing high-value actions. The agent also explores several new nearby states for which it does not have Q-estimates. Snapshot 4 shows the final Q-value estimates. A set of short videos of the GridWorlds environment, showing

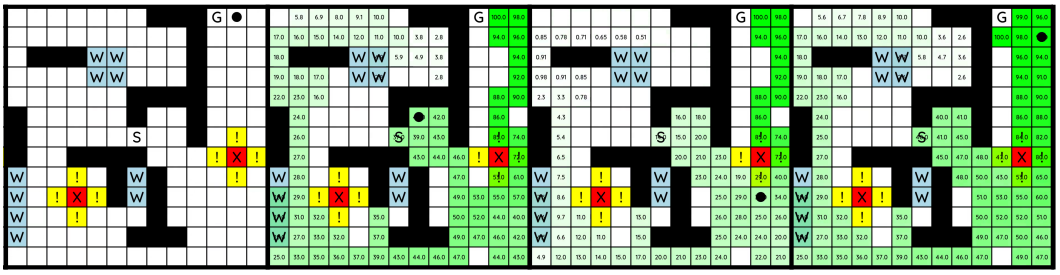


Figure 4: An RL^3 policy on a selected meta-episode visualized using a sequence of snapshots. 'S' is the starting tile, 'G' is the goal tile and the black circle shows the current position of the agent. Blue tiles marked 'W' are wet tiles. Wet tiles always lead to the agent slipping to one of the directions orthogonal to the intended direction of movement. Entering wet tiles yield an immediate reward of -2. Yellow tiles marked '!' are warning tiles and entering them causes -10 reward. Red tiles marked 'X' are fatally dangerous. Entering them ends the episode and leads to a reward of -100. Black tiles are obstacles. White tiles yield a reward of -1 to incentive the agent to reach the goal quickly. On all tiles other than wet tiles, there is a chance of slipping sideways with a probability of 0.2. The object-level state-values $v^t(s) = \max_a Q^t(s, a)$, as approximated by object-level RL, is represented using shades of green (and the accompanying text), where darker shades represent higher values.

both RL^2-transformer and RL^3 agents solving the same set of problem instances, can be found at https://youtu.be/eLA_S1BQUYM.

Computation Overhead Considerations: As mentioned earlier, for implementing object-level RL, we use model estimation followed by finite-horizon value-iteration to obtain Q-estimates. The computation overhead is negligible for Bandits (5 actions, task horizon = 1) and very little for the MDPs domain (10 states, 5 actions, task horizon 10). For 13x13 GridWorlds (up to 169 states, 5 actions, task horizon = 350), RL^3 takes approximately twice the computation time of RL^2-transformer per meta-episode. However, RL^3-coarse requires only 10% overhead while still outperforming RL^2-transformer and retaining more than 90% of the performance of RL^3. This demonstrates the utility of state abstractions in RL^3 for scaling. Finally, the meta-training sample efficiency demonstrated by RL^3 translates directly to wall-time efficiency as training is dominated by gradient computation, *not* value iteration during data collection in PPO. Our implementation is available at https://github.com/bhatiaabhinav/RL3.

7 Limitations and Conclusion

Though it compares favorably to strong Meta-RL approaches like RL^2-transformer where applicable, RL^3 does have some limitations. First, it assumes the object-level decision-making model is an MDP, which although a common assumption in the literature, may be challenged in practice. While in principle we could extend RL^3 to POMDPs using methods like point-based value iteration, this has yet to be tested empirically. Second, RL^3 relies on fast, potentially approximate methods for object-level RL, and using value iteration complicates application to problems with continuous state spaces. However, we speculate that a crude linear function approximation would suffice. Finally, inference time is slightly slower at deploy time due to running object-level RL. However, the overall training time is actually faster because of better meta-training efficiency. In fact, RL^3 could enable working with adaptation periods that are otherwise prohibitively long for many Meta-RL approaches.

To conclude, in this paper, we introduced RL^3, a principled hybrid approach that combines the strengths of traditional RL and Meta-RL and provides a more robust and efficient Meta-RL algorithm. We advanced intuitive and theoretical arguments regarding its suitability for Meta-RL and presented empirical evidence to validate those ideas. Specifically, we demonstrated that RL^3 holds potential to enhance long-term performance, generalization on out-of-distribution tasks and reducing meta-training time.

Acknowledgments

This work was supported in part by the National Science Foundation grant numbers 1954782, 2205153, and 2321786.

References

Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sk2u1g-0-.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Conference on Robot Learning*, pp. 1478–1487, 2022.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. URL https://aclanthology.org/D14-1179.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta learning of exploration. *arXiv preprint arXiv:2008.02598*, 2020.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Michael O'Gordon Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, 2002.

Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

David Emukpere, Xavier Alameda-Pineda, and Chris Reinke. Successor feature neural episodic control. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=e1Q2_jaE08J.

Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer Q-networks for partially observable reinforcement learning. In *NeurIPS Workshop on Foundation Models for Decision Making*, 2022. URL https://openreview.net/forum?id=DrzwyQZNJz.

Eyal Even-Dar, Yishay Mansour, and Peter Bartlett. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5(1), 2003.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-Q-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeD3CEFPH.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, pp. 1126–1135, 2017.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130, 2018.

Ali Ghadirzadeh, Xi Chen, Petra Poklukar, Chelsea Finn, Mårten Björkman, and Danica Kragic. Bayesian meta-learning for few-shot policy adaptation across robotic platforms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1274–1280, 2021.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483, 2015.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in Neural Information Processing Systems*, 31, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870, 2018.

Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Rituraj Kaushik, Timothée Anne, and Jean-Baptiste Mouret. Fast online adaptation in robotics through meta-learning embeddings of simulated priors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5269–5276, 2020.

Michael Kearns and Satinder Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems*, volume 11, 1998.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pp. 282–293, 2006.

Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. Meta reinforcement learning with task embedding and shared policy. In *International Joint Conference on Artificial Intelligence*, pp. 2794–2800, 2019.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International Conference on Machine Learning*, pp. 6925–6935, 2021.

Zhao Mandi, Pieter Abbeel, and Stephen James. On the effectiveness of fine-tuning versus meta-RL for robot manipulation. In *CoRL Workshop on Pre-training Robot Learning*, 2022. URL https://openreview.net/forum?id=21TVvjhOkV.

Luckeciano C Melo. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pp. 15340–15359, 2022.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1DmUzWAW.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free RL can be a strong baseline for many POMDPs. In *International Conference on Machine Learning*, pp. 16691–16723, 2022. URL https://proceedings.mlr.press/v162/ni22a.html.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. *arXiv preprint arXiv:1909.09157*, 2019.

Allen Z Ren, Bharat Govil, Tsung-Yen Yang, Karthik R Narasimhan, and Anirudha Majumdar. Leveraging language for accelerated learning of tool manipulation. In *Conference on Robot Learning*, pp. 1531–1541, 2023.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Csaba Szepesvári. The asymptotic convergence-rate of Q-learning. In *Advances in Neural Information Processing Systems*, volume 10, 1997.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2753–2762, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18:77–95, 2002.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. *Advances in Neural Information Processing Systems*, 32, 2019.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Jane X Wang, Michael King, Nicolas Pierre Mickael Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, H. Francis Song, Gavin Buttimore, David P Reichert, Neil Charles Rabinowitz, Loic Matthey, Demis Hassabis, Alexander Lerchner, and Matthew Botvinick. Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://openreview.net/forum?id=eZu4BZxlRnX.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Zheng Xiong, Luisa M Zintgraf, Jacob Austin Beck, Risto Vuorio, and Shimon Whiteson. On the practical consistency of meta-reinforcement learning algorithms. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=xwQgKphwhFA.

Liqi Yan, Dongfang Liu, Yaoxian Song, and Changbin Yu. Multimodal aggregation approach for memory vision-voice indoor navigation with meta-learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5847–5854, 2020.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pp. 7693–7702, 2019.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for Bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations*, 2020.

Luisa M Zintgraf, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson. Exploration in approximate hyper-state space for meta reinforcement learning. In *International Conference on Machine Learning*, pp. 12991–13001, 2021.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Proofs

A.1 Bayes Optimality of Q-value Estimates in Bernoulli Multi-armed Bandits

Given an instance of a Bernoulli multi-armed bandit MDP, $M_i \sim \mathcal{M}$, and trajectory data $\Upsilon_{1:T}$ up to time T, we would like to show that the probability $P(i|\Upsilon_{1:T})$ can be determined entirely from Q-estimates Q_i^T and action-counts N_i^T, as long as the initial belief is uniform or known.

In the following proof, we represent an instance i of K-armed Bandits as a K-dimensional vector of success probabilities $[p_{i1}, ..., p_{iK}]$, such that pulling arm k is associated with reward distribution $P(r = 1|i, k) = p_{ik}$ and $P(r = 0|i, k) = (1 - p_{ik})$.

Let the number of times arm k is pulled up to time T be N_{ik}^T, and the number of successes associated with pulling arm k up to time T be q_{ik}^T. Given that this is an MDP with just a single state and task horizon of 1, the Q-estimate associated with arm k is just the average reward for that action, which is the ratio of successes to counts associated with that action i.e., $Q_{ik}^T = \frac{q_{ik}^T}{N_{ik}^T}$. To reduce the clutter in the notation, we will drop the superscript T for the rest of the subsection. Now,

$$P(i|\Upsilon_{1:T}) = \alpha P(i) \cdot P(\Upsilon_{1:T}|i) \tag{7}$$

where α is the normalization constant, $P(i)$ is the prior probability of task i (which is assumed to be known beforehand), and $\Upsilon_{1:T}$ is the sequence of actions and the corresponding rewards up to time T. Assuming, without loss of generality, that the sequence of actions used to disambiguate tasks is a given, $P(\Upsilon_{1:T}|i)$ becomes simply the product of probabilities of reward outcomes up to time T, noting that the events are independent. Therefore,

$$\begin{aligned} P(\Upsilon_{1:T}|i) &= \prod_{k=1:K} \prod_{t=1:T} ([r_{tk} = 1]p_{ik} + [r_{tk} = 0](1 - p_{ik})) && (8) \\ &= \prod_{k=1:K} p_{ik}^{q_{ik}} \cdot (1 - p_{ik})^{N_{ik} - q_{ik}} && (9) \\ &= \prod_{k=1:K} p_{ik}^{Q_{ik} N_{ik}} \cdot (1 - p_{ik})^{N_{ik} - Q_{ik} N_{ik}} && (10) \end{aligned}$$

Putting everything together,

$$P(i|\Upsilon_{1:T}) = \alpha P(i) \cdot \prod_{k=1:K} p_{ik}^{Q_{ik} N_{ik}} \cdot (1 - p_{ik})^{N_{ik} - Q_{ik} N_{ik}} \tag{11}$$

This equation proves that N_i^T and Q_i^T are sufficient statistics to determine $P(i|\Upsilon_{1:T})$ in this domain, assuming that the prior over task distribution is known. □

A.2 Non-Bayes Optimality of Q-value Estimates in Gaussian Multi-armed Bandits

Given an instance of a Gaussian multi-armed bandit MDP, $M_i \sim \mathcal{M}$, and trajectory data $\Upsilon_{1:T}$ up to time t, here we derive the closed-form expression of the probability $P(i|\Upsilon_{1:T})$ and show that it contains terms other than Q-estimates Q_i^t and action-counts N_i^t.

In the following proof, we represent an instance i of K-armed Bandits as a $2K$-dimensional vector of means and standard deviations $[\mu_{i1}, ..., \mu_{iK}, \sigma_{i1}, ..., \sigma_{iK}]$, such that pulling arm k is associated with reward distribution $P(r|i, k) = \frac{1}{\sqrt{2\pi\sigma_{ik}}} \exp(\frac{r - \mu_{ik}}{\sigma_{ik}})^2$.

Let the number of times arm k is pulled up to time T be N_{ik}^T. Given this is an MDP with a single state and the task horizon is 1, the Q-estimate associated with arm k is just the average reward for

that action $\text{Avg}[r_k]$ up to time T. To reduce the clutter in the notation, we will drop the superscript T for the rest of the subsection. As in the previous subsection, we now compute the likelihood $P(\Upsilon_{1:T}|i)$.

$$P(\Upsilon_{1:T}|i) = \prod_{k=1:K} \prod_{t=1:T} \frac{1}{\sqrt{2\pi\sigma_{ik}}} \exp(\frac{r_{tk} - \mu_{ik}}{\sigma_{ik}})^2 \tag{12}$$

Therefore, the log likelihood is

$$\log P(\Upsilon_{1:T}|i) = \sum_{k=1:K} \sum_{t=1:T} \frac{(r_{tk} - \mu_{ik})^2}{\sigma_{ik}^2} - \log(2\pi\sigma_{ik})/2 \tag{13}$$

$$= \sum_{k=1:K} N_{ik} \frac{\text{Avg}[(r_{tk} - \mu_{ik})^2]}{\sigma_{ik}^2} - N_{ik} \log(2\pi\sigma_{ik})/2 \tag{14}$$

$$= \sum_{k=1:K} N_{ik} \frac{\text{Avg}[r_k^2] - 2\mu_{ik}\text{Avg}[r_k] + \mu_{ik}^2}{\sigma_{ik}^2} - N_{ik} \log(2\pi\sigma_{ik})/2 \tag{15}$$

$$= \sum_{k=1:K} N_{ik} \frac{(\text{Var}[r_k] + \text{Avg}[r_k]^2) - 2\mu_{ik}\text{Avg}[r_k] + \mu_{ik}^2}{\sigma_{ik}^2} - N_{ik} \log(2\pi\sigma_{ik})/2 \tag{16}$$

$$= \sum_{k=1:K} N_{ik} \frac{\text{Var}[r_k] + (Q_{ik})^2 - 2\mu_{ik}Q_{ik} + \mu_{ik}^2}{\sigma_{ik}^2} - N_{ik} \log(2\pi\sigma_{ik})/2 \tag{17}$$

Therefore, computing this expression requires computing the variance in rewards, $\text{Var}[r_k]$, associated with each arm up to time T, apart from the Q-estimates and action-counts. This proves that Q-estimates and action-counts alone are insufficient to completely determine $P(i|\Upsilon_{1:T})$ in Gaussian multi-armed bandits domain. □

A.3 Object-level Q-estimates and Meta-level Values

Proof of Equation 4. In standard Meta-RL, the POMDP state is $\bar{s}_t = [s_t, i]$, but the agent only observes the MDP state s_t, i.e., $\bar{\omega}_t = s_t$, while the task identity i remains hidden. In RL^3, however, $\bar{\omega}_t$ additionally includes the vector of Q-estimates $Q_i^t(s_t)$ for the hidden task. Consequently, the observation function is defined as $\bar{O}\big(\bar{\omega}_t \mid \bar{s}_t, a_{t-1}\big) = \Pr\big(\bar{\omega}_t = [s_t, Q_i^t(s_t)] \mid [s_t, i], a_{t-1}\big)$, encoding the probability that the observed Q-values are consistent with task i. Because the observation model appears in the belief update rule (see Section 3.1), and because Q-estimates provide strong evidence for task identification (Appendix D), robust recovery of the task belief is possible even with a noisy prior. Once the Q-estimates converge, consider the following two cases:

Case 1 (unique Q-signature). If the observed Q-values are unique to MDP M_i, the belief distribution rapidly collapses to zero probability for all tasks $j \neq i$, yielding $\bar{V}^*(\bar{b}) = \max_{a \in A} Q_i(s, a)$.

Case 2 (non-unique Q-signature). If several tasks share identical Q-values, the belief concentrates only on this subset of tasks. Since all remaining tasks induce the same Q-values, $\bar{V}^*(\bar{b})$ still simplifies to $\max_{a \in A} Q_i(s, a)$, where i denotes any of the tasks with non-zero belief.

This establishes Equation 4. In the limit, after all state-action pairs have been explored, the task identity can be inferred exactly, reducing the meta-level value function to the optimal object-level value. However, the critical insight of Equation 4 is that RL^3 achieves this equivalence directly through Q-based observations, without relying on the full experience stream. □

Proof of Equation 6. The optimal meta-level value function in belief space satisfies the Bellman equation:

$$\bar{V}^*(\bar{b}) = \max_{a \in A} \Big[\sum_{\bar{s} \in \bar{S}} \bar{b}(\bar{s}) \bar{R}(\bar{s}, a) + \gamma \sum_{\bar{\omega} \in \bar{\Omega}} \Pr(\bar{\omega} | \bar{b}, a) \bar{V}^*(\bar{b}') \Big]. \tag{18}$$

Because the only hidden variable in $\bar{s} = [s, i]$ is the task identity i, this expression reduces to

$$\bar{V}^*(\bar{b}) = \max_{a \in A} \Big[\sum_{M_i \in \mathcal{M}} \bar{b}(i) R_i(s, a) + \gamma \sum_{\bar{\omega} \in \bar{\Omega}} \Pr(\bar{\omega} | \bar{b}, a) \bar{V}^*(\bar{b}') \Big], \tag{19}$$

where the observation likelihood is computed as $\Pr(\bar{\omega}|\bar{b}, a) = \sum_{\bar{s}'} \bar{O}(\bar{\omega}|\bar{s}', a) \sum_{\bar{s}} \Pr(\bar{s}'|\bar{s}, a) \cdot \bar{b}(\bar{s})$. □

B Architecture

B.1 RL^2

Our modified implementation of RL^2 uses transformer decoders (Vaswani et al., 2017) instead of RNNs to map trajectories to action probabilities and meta-values, in the actor and the critic, respectively, and uses PPO instead of TRPO for outer RL. The decoder architecture is similar to (Vaswani et al., 2017), with 2 layers of masked multi-headed attention. However, we use learned position embeddings instead of sinusoidal, followed by layer normalization. Our overall setup is similar to (Esslinger et al., 2022).

For each meta-episode of interactions with an MDP M_i, the actor and the critic transformers look at the entire history of experiences up to time t and output the corresponding action probabilities $\pi_1 ... \pi_t$ and meta-values $\bar{V}_1 ... \bar{V}_t$, respectively. An experience input to the transformer at time t consists of the previous action a_{t-1}, the latest reward r_{t-1}, the current state s_t, episode time step t_τ, and the meta-episode time step t, all of which are normalized to be in the range $[0, 1]$. In order to reduce inference complexity, say at time step t, we append t new attention scores (corresponding to experience input t w.r.t. the previous $t-1$ experience inputs) to a previously cached $(t-1) \times (t-1)$ attention matrix, instead of recomputing the entire $t \times t$ attention matrix. This caching mechanism is implemented for each attention head and reduces the inference complexity at time t from $\mathcal{O}(t^2)$ to $\mathcal{O}(t)$.

B.2 RL^3

The input of the transformer in RL^3 includes a vector of Q estimates (in practice, they are supplied as the vector of advantage estimates ($Q - \max_a Q$) along with the value function ($\max_a Q$) separately) and a vector of action counts at each step t for the corresponding state. As mentioned in Section 4.2, this is implemented in our code simply by converting MDPs in the problem set to VAMDPs using a wrapper and running our implementation of RL^2 thereafter. The pseudocode is shown in the algorithm 1. The Markov version of RL^3 uses a dense neural network, with two hidden layers of 64 nodes each, with the ReLU activation function.

For object-level RL, we use model estimation followed by value iteration (with discount factor $\gamma = 1$) to obtain Q-estimates. The transition probabilities and the mean rewards are estimated using maximum likelihood estimation (MLE), with Laplace smoothing (coefficient = 0.1) for transition probabilities estimation. For unseen actions, rewards are assumed to be zero, and transitions equally likely to other states. States are added to the model incrementally when they are visited, so that value iteration does not compute values for unvisited states. Moreover, value iteration is carried out only for iterations equal to the task horizon (which is 1, 10, 250, 350 for Bandits, MDPs, 11x11 GridWorlds, 13x13 GridWorlds domains, respectively), unless the maximum Bellman error drops below 0.01.

Algorithm 1 Value-Augmenting Wrapper for Discrete MDPs

procedure RESETMDP(vamdp)
  vamdp.$t \leftarrow 0$; vamdp.$t_\tau \leftarrow 0$
  vamdp.$N[s,a] \leftarrow 0$; vamdp.$Q[s,a] \leftarrow 0 \quad \forall s \in S, a \in A$
  vamdp.rl $\leftarrow$ INITRL()
  $s =$ RESETMDP(vamdp.mdp)
  return ONEHOT$(s) \cdot Q[s] \cdot N[s]$
procedure STEPMDP(vamdp, a)
  $s \leftarrow$ mdp.s
  $r, s' \leftarrow$ STEPMDP(vamdp.mdp, a)
  $d \leftarrow$ TERMINATED(vamdp.mdp)
  vamdp.t, vamdp.$N[s,a]$, vamdp.$t_\tau \leftarrow += 1$
  vamdp.$Q \leftarrow$ UPDATERL(vamdp.rl, s, a, r, s', d)
  if d **or** vamdp.$t_\tau \geq$ `task_horizon` **then**
    vamdp.$t_\tau \leftarrow 0$
    $s' \leftarrow$ RESETMDP(vamdp.mdp)
  return r, ONEHOT$(s') \cdot Q[s'] \cdot N[s']$ ▷ Concatenate state, Q-estimates and action counts
procedure TERMINATED(vamdp)
  return vamdp.$t \geq H$

B.3 RL^3-coarse

During model estimation in RL^3-coarse, concrete states in the underlying MDP are incrementally clustered into abstract states as they are visited. When a new concrete state is encountered, its abstract state ID is set to that of a previously visited state within a 'clustering radius', unless that previous state is already part of a full cluster (determined by a maximum 'cluster size' parameter). If multiple visited states satisfy the criteria, the ID of the closet one is chosen. If none of the visited states that satisfy the criteria, then the new state is assigned a new abstract state ID, increasing the number of abstract states in the model. It is worth noting that this method of deriving abstractions does not take advantage of any structure in the underlying domain. However, this simplicity makes it general purpose, efficient, and impartial, while still leading to excellent performance. For our GridWorlds domain, we chose a cluster size of 2 and a clustering radius such that only non-diagonal adjacent states are clustered (Manhattan radius of 1).

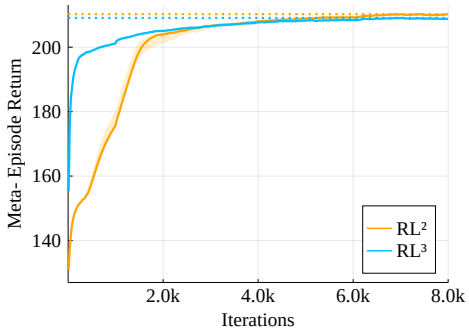
The mechanism for learning the transition function and the reward function in the abstract MDP is the same as before. For estimating Q-values for a given concrete state, value iteration is carried out on the abstract MDP and the Q-estimates of the corresponding abstract state are returned.

C Training

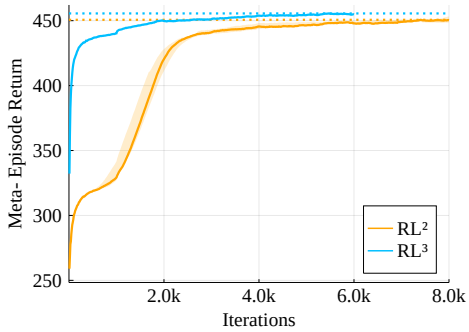
Figs. 5, and 6 show the training curves for MDPs, and GridWorlds environments, respectively, across 3 random seeds. The results in the main text correspond to the median model. We ran the experiments on Nvidia GeForce RTX 2080 Ti GPUs for context length ≤ 256 which took approximately 12-24 hours, and on Nvidia A100 GPUs for higher context lengths, which took 1-2 days.

D Additional Analysis

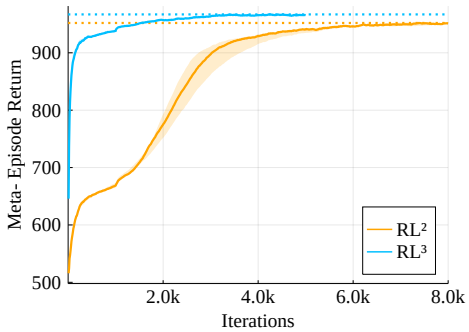
In this section, we show that Q-estimates, though imperfect, produce reasonable signals for task identification. Here, we test this claim thoroughly with 3 analyses.



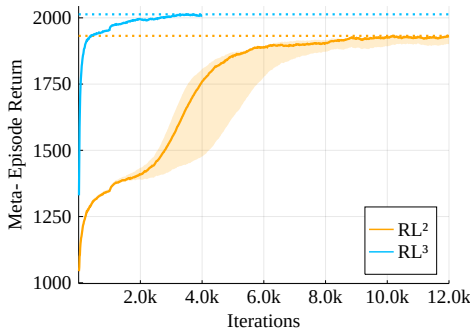
(a) $H = 128$



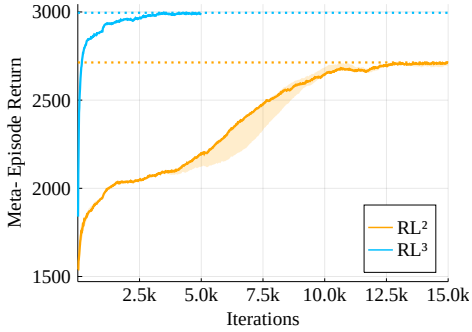
(b) $H = 256$



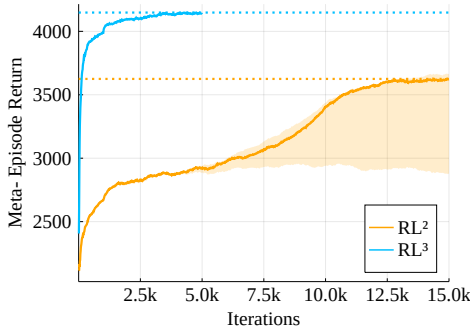
(c) $H = 512$



(d) $H = 1024$



(e) $H = 1500$



(f) $H = 2048$

Figure 5: Average meta-episode return vs PPO iterations for MDPs domain for different interaction budgets.

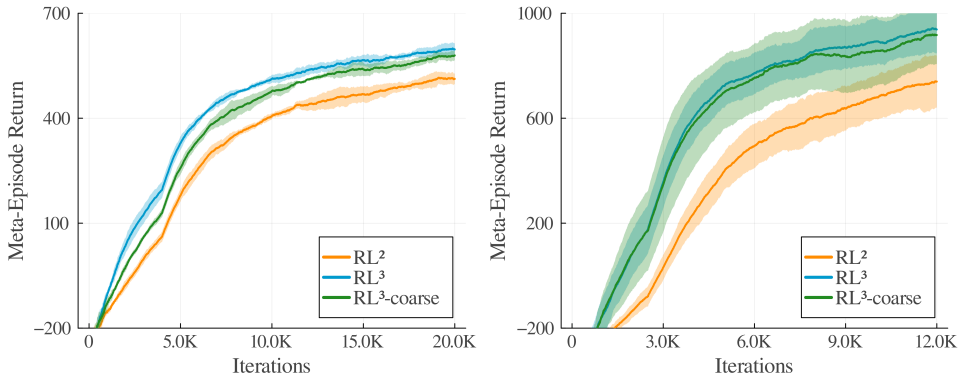


Figure 6: Average meta-episode return vs PPO iterations for GridWorlds 11x11 (*left*) and 13x13 (*right*).

D.1 Requirements for a Unique Q^*-Function

Throughout, we assume fixed state space and action space. Below, we show that if the transition function is fixed, then two Q^*-tables will be identical if and only if both reward functions are also equal. First, we show that identical Q^* functions imply identical reward functions. Given the Bellman equations,

$$Q_1^*(s,a) = R_1(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q_1^*(s',a') \tag{20}$$

$$Q_2^*(s,a) = R_2(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q_2^*(s',a') \tag{21}$$

Substituting $Q_2^* = Q_1^*$ in Equation (21), we get

$$Q_1^*(s,a) = R_2(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q_1^*(s',a') \tag{22}$$

Subtracting Equation (20) from Equation (22), we get $R_1(s,a) = R_2(s,a)$. Thus, $(Q_1^* = Q_2^*) \wedge (T_1 = T_2) \implies (R_1 = R_2)$.

Now, if two MDPs have the same reward and transition function, they are the same MDP and will have the same optimal value function. So, $(R_1 = R_2) \wedge (T_1 = T_2) \implies (Q_1^* = Q_2^*)$.

Since encountering similar Q^*-tables is thus dependent on both transitions and rewards 'balancing' each other, the question is then for practitioners: How likely are we to get many MDPs that all appear to have very similar Q^*-tables?

D.2 Empirical Test using Max Norm

Given an MDP with 3 states and 2 actions, we want to find the probability that $||Q_1^* - Q_2^*||_\infty < \delta$, where Q_1^* and Q_2^* are 6-entry (3 states $\times$ 2 actions) Q^*-tables. The transition and reward functions are drawn from distributions parameterized by α and β, respectively. Transition probabilities are drawn from a Dirichlet distribution, Dir(α), and rewards are sampled from a normal distribution, $\mathcal{N}(1, \beta)$. In total, we ran 3 combinations of α and β, each with 50,000 MDPs, a task horizon of 10, and $\delta = 0.1$. To get the final probability, we test all $((50,000 - 1)^2)/2$ non-duplicate pairs and count the number of max norms less than δ.

Results: For $\alpha = 1.0$, $\beta = 1.0$, we found the probability of a given pair of MDPs having duplicate Q^*-table to be $\epsilon = 2.6 \times 10^{-9}$. For $\alpha = 0.1$, $\beta = 1.0$, which is a more deterministic setting, we found $\epsilon = 4.6 \times 10^{-9}$. Further, with $\alpha = 0.1$, $\beta = 0.5$, where rewards are more closely distributed,

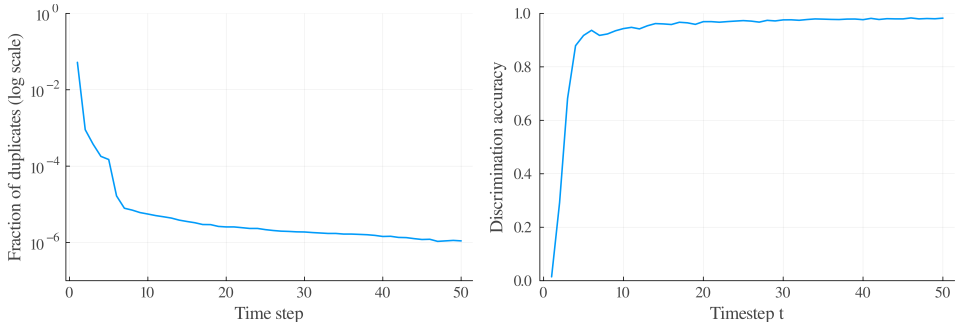


Figure 7: The task-identification power of Q-estimates. *Left*: Fraction of δ-duplicates, with $\delta = 0.1$, as a function of time steps in a set of 5,000 random MDPs. *Right*: Accuracy of a simple multi-class classifier in predicting task ID given Q-table estimates, as function of time step. Both figures are generated using the same policy.

we found $\epsilon = 1.1 \times 10^{-7}$. Overall, we can see that even for a set of very small MDPs, the probability of numerically mistaking one Q^*-table for another is vanishingly small.

D.3 Predicting Task Families

The near uniqueness of Q^*-functions is encouraging, but max norm is not a very sophisticated metric. Here, we test whether a very simple multi-class classifier (1 hidden layer of 64 nodes), can accurately identify individual tasks based on their *Q-estimates*. Moreover, we track how the classification accuracy improves as a function of the number of steps taken within the MDP as the estimates improve. In this experiment, the same random policy is executed in each MDP for 50 time steps. As before, our MDPs have 3 states and 2 actions.

We instantiate 10,000 MDPs whose transition and reward functions are drawn from the same distribution as before: transitions from a Dirichlet distribution with $\alpha = 0.1$ and rewards sampled from a normal distribution $N(1, 0.5)$. Thus, this is a classification problem with 10,000 classes. *A priori*, this exercise seems relatively difficult given the number of tasks and the parameters chosen for the distributions. Fig. 7 shows a compelling result given the simplicity of the model and the relative difficulty of the classification problem. Clearly, Q-estimates, even those built from only 20 experiences, provide a high signal-to-noise ratio w.r.t. task identification. And this is for a *random* policy. In principle, the Meta-RL agent could follow a much more deliberate policy that actively disambiguates trajectories such that the Q-estimates evolve in a way that leads to faster or more reliable discrimination.

E Domain Descriptions

E.1 Bernoulli Multi-Armed Bandits

We use the same setup described by Duan et al. (2016). At the beginning of each meta-episode, the success probability corresponding to each arm is sampled from a uniform distribution $\mathcal{U}(0, 1)$. To test OOD generalization, we sample success probabilities from $\mathcal{N}(0.5, 0.5)$

E.2 Random MDPs

We use the same setup described by Duan et al. (2016). The MDPs have 10 states and 5 actions. For each meta-episode, the mean rewards $R(s, a)$ and transition probabilities $T(s, a, s')$ are initialized from a normal distribution ($\mathcal{N}(1, 1)$) and a flat Dirichlet distribution ($\alpha = 1$), respectively. Moreover, when an action a is performed in state s, a reward is sampled from $\mathcal{N}(R(s, a), 1)$. To test OOD generalization, the transition probabilities are initialized with Dirichlet $\alpha = 0.25$.

Each episode begins at state $s = 1$ and ends after `task_horizon` = 10 time steps.

E.3 GridWorlds

A set of navigation tasks in a 2D grid environment. We experiment with 11x11 (121 states) and 13x13 (169 states) grids. The agent always starts in the center of the grid and needs to navigate through obstacles to a single goal location. The goal location is always at a minimum of `min goal manhat` Manhattan distance from the starting tile. The grid also contains slippery wet tiles, fatally dangerous tiles and warning tiles surrounding the latter. There are `num obstacle sets` set of obstacles, and each obstacle set spans `obstacle set len` tiles, in either horizontal or vertical configuration. There are `num water sets` set of wet regions and each wet region always spans `water set length`, in either a horizontal or vertical configuration. Entering wet tiles yields an immediate reward of -2. There are `num dangers` danger tiles and entering them ends the episode and leads to a reward of -100. Warning tiles always occur as a set of 4 tiles non-diagonally surrounding the corresponding danger tiles. Entering warning tiles causes -10 reward. Normal tiles yield a reward of -1 to incentivize the agent to reach the goal quickly. On all tiles, there is a chance of slipping sideways with a probability of 0.2, except for wet tiles, where the probability of slipping sideways is 1.

The parameters for our canonical 11x11 and 13x13 GridWorlds are as follows: `num obstacle sets` = 11, `obstacle set len` = 3, `num water sets` = 5, `water set length` = 2, `num dangers` = 2, and `min goal manhat` = 8. The parameters for the OOD variations are largely the same and the differences are as follows. For DETERMINISTIC variation, the slip probability on non-wet tiles is 0. For DENSE variation, `obstacle set len` is increased to 4. For WATERY variation, `num water sets` is increased to 8. For DANGEROUS variation, `num dangers` is increased to 4. For CORNER variation, `min goal manhat` is set to 12, so that the goal is placed on one of the corners of the grid.

There is no fixed task horizon for this domain. An episode ends when the agent reaches the goal or encounters a danger tile. In principle, an episode can last through the entire meta-episode if a terminal state is not reached.

When a new grid is initialized at the beginning of each meta-episode, we ensure that the optimal, non-discounted return within a fixed horizon of 100 steps is between 50 and 100. This is to ensure that the grid both has a solution and the solution is not trivial.

Table 3: RL^2 /RL^3 Hyperparameters

Hyperparameter	**Value**
Learning Rate (Actor and Critic)	0.0003 (Bandits, MDPs)
	0.0002 (GridWorlds)
Adam $\beta 1, \beta 2, \epsilon$	$0.9, 0.999, 10^{-7}$
Weight Decay (Critic Only)	10^{-2}
Batch size	32768
Rollout Length	Interaction Budget (H)
Number of Parallel Environments	Batch Size $\div$ H
Minibatch Size	4096
Entropy Regularization Coefficient	0.1 with decay (MDPs)
	0.04 (GridWorlds)
	0.01 (Bandits)
PPO Iterations	See training curves
Epochs Per Iteration	8
Max KL Per Iteration	0.01
PPO Clip ϵ	0.2
GAE λ	0.3
Discount Factor γ	0.99
Decoder Layers	2
Attention Heads	4
Activation Function	GELU
Decoder Size (d_model)	64