

Foundations and Trends® in Machine Learning

A Tutorial on Meta-Reinforcement Learning

Suggested Citation: Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn and Shimon Whiteson (2025), “A Tutorial on Meta-Reinforcement Learning”, Foundations and Trends® in Machine Learning: Vol. 18, No. 2-3, pp 224–384. DOI: 10.1561/2200000080.

Jacob Beck

University of Oxford

jacob.beck@cs.ox.ac.uk

Risto Vuorio

University of Oxford

risto.vuorio@cs.ox.ac.uk

Evan Zheran Liu

Stanford University

evanliu@cs.stanford.edu

Zheng Xiong

University of Oxford

zheng.xiong@cs.ox.ac.uk

Luisa Zintgraf

University of Oxford

zintgraf@deepmind.com

Chelsea Finn

Stanford University

cbfinn@cs.stanford.edu

Shimon Whiteson

University of Oxford

shimon.whiteson@cs.ox.ac.uk

This article may be used only for the purpose of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval.

now

the essence of knowledge

Boston — Delft

Contents

1	Introduction	226
2	Background	230
2.1	Reinforcement Learning	230
2.2	Meta-RL Definition	231
2.3	POMDP Formalization	235
2.4	Example Algorithms	238
2.5	Problem Categories	243
2.6	Related Work	245
3	Few-shot Meta-RL	248
3.1	Parameterized Policy Gradient Methods	252
3.2	Black Box Methods	260
3.3	Task Inference Methods	267
3.4	Exploration and Meta-Exploration	277
3.5	Bayes-adaptive Optimality	286
3.6	Supervision	291
3.7	Model-based Meta-RL	300
3.8	Theory of Meta-RL	303

4	Many-shot Meta-RL	307
4.1	Multi-task Many-shot Meta-RL	309
4.2	Single-task Many-shot Meta-RL	311
4.3	Many-shot Meta-RL Methods	313
5	Applications	320
5.1	Robotics	321
5.2	Multi-agent RL	324
6	Open Problems	329
6.1	Few-shot Meta-RL Generalization	329
6.2	Many-shot Meta-RL: Optimization Issues and Standard Benchmarks	334
6.3	Utilizing Offline Data in Meta-RL	336
6.4	Limitations	339
7	Conclusion	342
Appendix		344
References		346

A Tutorial on Meta-Reinforcement Learning

Jacob Beck^{1*}[†], Risto Vuorio^{1*}[‡], Evan Zheran Liu^{2◊}, Zheng Xiong¹, Luisa Zintgraf^{1§}, Chelsea Finn² and Shimon Whiteson¹

¹University of Oxford, UK; *jacob.beck@cs.ox.ac.uk, risto.vuorio@cs.ox.ac.uk, zheng.xiong@cs.ox.ac.uk, zintgraf@deepmind.com, shimon.whiteson@cs.ox.ac.uk*
²Stanford University, USA; *evanliu@cs.stanford.edu, cbfinn@cs.stanford.edu*

ABSTRACT

While deep reinforcement learning (RL) has fueled multiple high-profile successes in machine learning, it is held back from more widespread adoption by its often poor data efficiency and the limited generality of the policies it produces. A promising approach for alleviating these limitations is to cast the development of better RL algorithms as a machine learning problem itself in a process called meta-RL. Meta-RL is most commonly studied in a problem setting where, given a distribution of tasks, the goal is to learn a policy that is capable of adapting to any new task from the task distribution with as little data as possible. In this survey, we describe the meta-RL problem setting in detail as well as its major variations. We discuss how, at a high level, meta-RL

*Contributed equally.

[†]Now at Oracle.

[‡]Now at Inflection.

[◊]Now at Imbue.

[§]Now at DeepMind.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn and Shimon Whiteson (2025), “A Tutorial on Meta-Reinforcement Learning”, Foundations and Trends® in Machine Learning: Vol. 18, No. 2-3, pp 224–384. DOI: 10.1561/2200000080.

©2025 J. Beck *et al.*

research can be clustered based on the presence of a task distribution and the learning budget available for each individual task. Using these clusters, we then survey meta-RL algorithms and applications. We conclude by presenting the open problems on the path to making meta-RL part of the standard toolbox for a deep RL practitioner.

1

Introduction

Meta-reinforcement learning (meta-RL) considers a family of machine learning (ML) methods that *learn to reinforcement learn*. That is, meta-RL methods use sample-inefficient ML to learn sample-efficient RL algorithms, or components thereof. As such, meta-RL is a special case of meta-learning (Vanschoren, 2018; Hospedales *et al.*, 2020; Huisman *et al.*, 2021), with the property that the learned algorithm is an RL algorithm. Meta-RL has been investigated as a machine learning problem for a significant period of time (Schmidhuber, 1987; Schmidhuber *et al.*, 1997; Thrun and Pratt, 1998; Schmidhuber, 2007). Intriguingly, research has also shown an analogue of meta-RL in the brain (Wang *et al.*, 2018).

Meta-RL has the potential to overcome some limitations of existing human-designed RL algorithms. While there has been significant progress in deep RL over the last several years, with success stories such as mastering the game of Go (Silver *et al.*, 2016), stratospheric balloon navigation (Bellemare *et al.*, 2020), or robot locomotion in challenging terrain (Miki *et al.*, 2022). RL remains highly sample inefficient, which limits its real-world applications. Meta-RL can produce (components of) RL algorithms that are much more sample efficient than existing RL methods, or even provide solutions to previously intractable problems.

At the same time, the promise of improved sample efficiency comes with two costs. First, meta-learning requires significantly more data than standard learning, as it trains an entire learning algorithm (often across multiple tasks). Second, meta-learning fits a learning algorithm to meta-training data, which may reduce its ability to generalize to other data. The trade-off that meta-learning offers is thus improved sample efficiency at test time, at the expense of sample efficiency during training and generality at test time.

Example application Consider, as a conceptual example, the task of automated cooking with a robot chef. When such a robot is deployed in somebody’s kitchen, it must learn a kitchen-specific policy, since each kitchen has a different layout and appliances. This challenge is compounded by the fact that not all items needed for cooking are in plain sight; pots might be tucked away in cabinets, spices could be stored on high shelves, and utensils might be hidden in drawers. Therefore, the robot needs not only to understand the general layout but also remember where specific items are once discovered. Training the robot directly in a new kitchen from scratch is too time consuming and potentially dangerous due to random behavior early in training. One alternative is to pre-train the robot in a *single* training kitchen and then fine-tune it in the new kitchen. However, this approach does not take into account the subsequent fine-tuning procedure. In contrast, meta-RL would train the robot on a *distribution* of training kitchens such that it can adapt to any new kitchen in that distribution. This may entail learning some parameters to enable better fine-tuning, or learning the entire RL algorithm that will be deployed in the new kitchen. A robot trained this way can both make better use of the data collected and also collect better data, e.g., by focusing on the unusual or challenging features of the new kitchen. This meta-learning procedure requires more samples than the simple fine-tuning approach, but it only needs to occur once, and the resulting adaptation procedure can be significantly more sample efficient when deployed in the new test kitchen.

This example illustrates how, in general, meta-RL may be particularly useful when the need for efficient adaptation is frequent, so the cost of meta-training is relatively small. This includes, but is not

limited to, safety-critical RL domains, where efficient data collection is necessary and exploration of novel behaviors is prohibitively costly or dangerous. In many cases, a large investment of sample-inefficient learning upfront (either with oversight, in a laboratory, or in simulation) is worthwhile to enable subsequent improved adaptation behavior. This example represents an aspirational application for meta-RL. In practice, meta-RL is applied to more limited robotics tasks such as robotic manipulation (Akkaya *et al.*, 2019; Zhao *et al.*, 2022b) and locomotion (Song *et al.*, 2020b).

Survey scope This is a survey of the meta-RL topic in machine learning and leaves out research on meta-RL in other fields such as neuroscience. Research on closely related machine learning topics is discussed in Section 2.6. To capture the breadth and depth of machine learning research on meta-RL, we surveyed the proceedings of several major machine learning conferences, as well as specialized workshops from the time period between 2017 and 2022. We found that a major portion of the meta-RL literature emerged post-2016, with the lion’s share of contributions being concentrated in three conferences: NeurIPS, ICML, and ICLR. For a full list of conferences and workshops covered, see Appendix A. While our survey primarily emphasizes these conferences and the specified timeframe, we also discuss a selection of relevant papers from outside this scope. From the proceedings of these conferences and workshops, we searched for papers that explicitly mention meta-RL as well those that do not make an explicit reference but that we judged to nevertheless fit the topic. Finally, we do not claim exhaustive coverage of meta-RL research included in our survey scope but rather a holistic overview of the most salient ideas and general directions.

Survey overview The aim of this survey is to provide an entry point to meta-RL, as well as reflect on the current state of the field and open areas of research. In Section 2, we define meta-RL and the different problem settings it can be applied to, together with two example algorithms.

In Section 3, we consider the most prevalent problem setting in meta-RL: few-shot meta-RL. Here, the goal is to learn an RL algorithm capable of *fast adaptation*, i.e., learning a task within just a handful of episodes.

These algorithms are often trained on a given task distribution, and meta-learn how to efficiently adapt to any task from that distribution. A toy example to illustrate this setting is shown in Figure 1.1. Here, an agent is meta-trained to learn how to navigate to different (initially unknown) goal positions on a 2D plane. At meta-test time, this agent can adapt efficiently to new tasks with unknown goal positions.

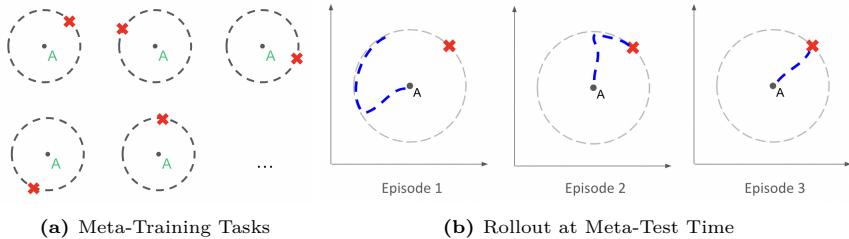


Figure 1.1: Example of the *fast adaptation* meta-RL problem setting discussed in Section 3. The agent (A) is meta-trained on a distribution of meta-training tasks to *learn* to go to goal position (X) located on a unit circle around its starting position (a). At meta-test time, the agent can adapt quickly (within a handful of episodes) to new tasks with initially unknown goal positions (b). In contrast, a standard RL algorithm may need hundreds of thousands of environment interactions when trained from scratch on one such task.

In Section 4, we consider many-shot settings. The goal here is to learn general-purpose RL algorithms not specific to a narrow task distribution, similar to those currently used in practice. There are two flavors of this: training on a distribution of tasks as above, or training on a single task but meta-learning alongside standard RL training.

Next, Section 5 presents some applications of meta-RL such as robotics. To conclude the survey, we discuss open problems in Section 6. These include generalization to broader task distributions for few-shot meta-RL, optimization challenges in many-shot meta-RL, and reduction of meta-training costs.

To provide high-level summaries of meta-RL research cited in this survey, we collect representative papers discussed in each section in a summary table presented within the section.

2

Background

Meta-RL can be broadly described as learning a part or all of an RL algorithm. In this section, we define and formalize meta-RL. In order to do so, we start by defining RL.

2.1 Reinforcement Learning

An RL algorithm learns a policy to take actions in a Markov decision process (MDP), also called the agent’s *environment*. An MDP is defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, P_0, R, \gamma, N \rangle$, where \mathcal{S} is the set of states, \mathcal{A} the set of actions, $P(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}_+$ the transition function, $P_0(s_0) : \mathcal{S} \rightarrow \mathbb{R}_+$ is a distribution over initial states, $R(s_t, a_t) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function, $\gamma \in [0, 1]$ is a discount factor, and N is the horizon. We give the definitions assuming a finite horizon for simplicity, though many of the algorithms we consider also work in the variable and infinite horizon setting. A policy is a function $\pi(a|s) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_+$ that maps states to action probabilities. The interaction of the policy with the MDP takes place in *episodes*, which start from initial states sampled from P_0 followed by N transitions between states where the actions are sampled from the policy π and states from the dynamics P . After the N transitions, a new episode begins starting from a freshly

sampled initial state. The reward for each transition is defined by the reward function $r_t = R(s_t, a_t)$. This defines a distribution over episodes

$$P(\tau) = P_0(s_0) \prod_{t=0}^{N-1} \pi(a_t|s_t)P(s_{t+1}|s_t, a_t). \quad (2.1)$$

We refer to the data $\tau = (s_0, a_0, r_0, \dots, s_{N-1}, a_{N-1}, r_{N-1}, s_N)$ collected during an episode as a *trajectory*. Note that the trajectory consists of one fewer action and reward than state, since the agent does not take an action in the final state.

The objective of RL is to learn a policy that maximizes the expected discounted return within an episode,

$$J(\pi) = \mathbb{E}_{\tau \sim P(\tau)} \left[\sum_{t=0}^{N-1} \gamma^t r_t \right], \quad (2.2)$$

where r_t are the rewards along the trajectory τ . In the process of optimizing this objective, multiple episodes are gathered. We denote the trajectories collected so far, across multiple episodes, including the current (incomplete) trajectory, as \mathcal{D} . When learning is done and \mathcal{D} is at its largest size, we write this as $\mathcal{D} = (\tau_0, \tau_1, \dots, \tau_H)$, where H is the maximum number of episodes used for learning.

An RL algorithm is a function that maps the data to a policy. This includes choosing the policies used for data collection during training. These intermediate policies are not necessarily greedy with respect to the RL objective but may take exploratory actions instead. In this survey, we mostly consider parameterized policies with parameters $\phi \in \Phi$. Therefore, we define an RL algorithm as the function $\phi = f(\mathcal{D})$.

2.2 Meta-RL Definition

RL algorithms are traditionally designed, engineered, and tested by humans. The idea of meta-RL is instead to learn (parts of) an algorithm f using machine learning. Where RL learns a policy, meta-RL learns the RL algorithm f that outputs the policy. This does not remove all of the human effort from the process, but shifts it from directly designing and implementing the RL algorithms into developing the

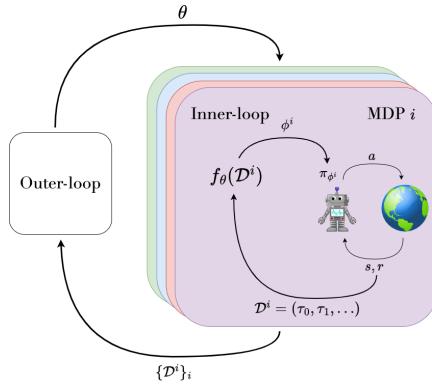


Figure 2.1: The relationship between the inner-loop and outer-loop in a meta-RL algorithm. The policy for MDP i , parameterized by ϕ^i , produces the meta-trajectory data \mathcal{D}^i . The inner-loop f_θ computes adapted policy parameters for each MDP based on the meta-trajectory during the policy’s interaction with the MDP. To compute the adapted parameters, the inner-loop can use all data collected in the MDP so far. The outer-loop computes updated meta-parameters using all of the meta-trajectories collected in all of the MDPs.

training environments and parameterizations required for learning parts of them in a data-driven way.

Due to this bi-level structure, the algorithm for learning f is often referred to as the *outer-loop*, while the learned f is called the *inner-loop*. The relationship between the inner-loop and the outer-loop is illustrated in Figure 2.1. Since the inner-loop and outer-loop both perform learning, we refer to the inner-loop as performing *adaptation* and the outer-loop as performing *meta-training*, for the sake of clarity. The learned inner-loop, that is, the function f , is assessed during *meta-testing*. We want to meta-learn an RL algorithm, or an inner-loop, that can adapt quickly to a new MDP. This meta-training requires access to a set of training MDPs. These MDPs, also known as *tasks*, come from a distribution denoted $p(\mathcal{M})$. In principle, the task distribution can be supported by any set of tasks. However, in practice, it is common for \mathcal{S} and \mathcal{A} to be shared between all of the tasks and the tasks to only differ in the reward $R(s, a)$ function, the dynamics $P(s'|s, a)$, and initial state distributions $P_0(s_0)$. Meta-training proceeds by sampling a task from the task distribution, running the inner-loop on it, and optimizing the

algorithm to improve the policies it produces. The interaction of the inner-loop with the task, during which the adaptation happens, is called a *lifetime* or a *trial* and is illustrated in Figure 2.2. A trial can consist of multiple episodes. After an episode ends, a new one begins with the initial state of the new episode sampled from the initial state distribution $P_0(s_0)$. The concatenation of all the data during a single trial, \mathcal{D} , which may contain multiple episodes, is called a *meta-trajectory*.

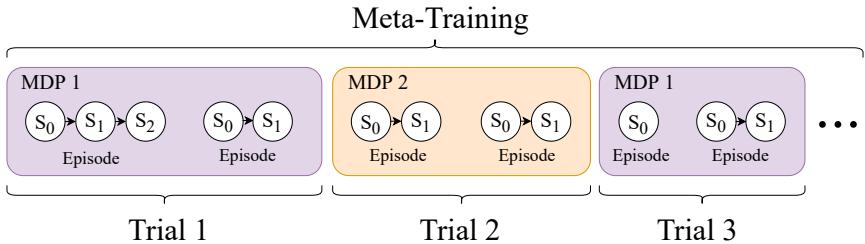


Figure 2.2: Meta-training consists of trials (or lifetimes), each broken up into multiple episodes from a single task (MDP). In this example, each trial consists of two episodes ($H = 2$).

Parameterization We generally parameterize the inner-loop f_θ with *meta-parameters* θ , and learn these parameters to maximize a meta-RL objective. Hence, f_θ outputs the parameters of π_ϕ directly: $\phi = f_\theta(\mathcal{D})$. We refer to the policy π_ϕ as the *base policy*. Frequently, the inner-loop will only adapt a subset of ϕ , which we refer to as the *adapted policy parameters*, or the *task parameters*, since they are adapted to each task. Some meta-RL algorithms use f_θ to map \mathcal{D} to ϕ at every MDP step, while others do so less frequently. In either case, we use ϕ_j to denote the j th set of parameters. Additionally, if the parameters are bound to a specific task, and there are a finite number of tasks, we use ϕ_j^i to indicate the j th set of parameters for the i th task. Likewise, we use \mathcal{D}_j^i to indicate the data for the j th update for the i th task.

Meta-RL objective In standard RL, a Markov policy suffices to maximize the RL objective given by Equation 2.2. In contrast, in meta-RL,

the inner-loop is an entire RL algorithm that outputs parameters of policies as a function of the meta-trajectory. The performance of a meta-RL algorithm is measured in terms of the returns achieved by the policies π_ϕ produced by the inner-loop during a trial on tasks \mathcal{M} drawn from the task distribution. Depending on the application, slightly different objectives are considered. For some applications, we can afford a free exploration or adaptation period, during which the performance of the policies produced by the inner-loop is not important as long as the final policy found by the inner-loop solves the task. The episodes during this phase can be used by the inner-loop for freely exploring the task. For other applications, a free exploration period is not possible and correspondingly the agent must maximize the expected return from the first timestep it interacts with the environment. Maximizing these different objectives leads to different learned exploration strategies: a free exploration period enables more risk-taking at the cost of wasted training resources when the risks are realized. Formally, the meta-RL objective for both settings is:

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{M} \sim p(\mathcal{M})} \left[\mathbb{E}_{\mathcal{D}} \left[G(\mathcal{D}) \middle| \pi_{f_\theta(\mathcal{D})}, \mathcal{M} \right] \right], \quad (2.3)$$

where $G(\mathcal{D}) = \sum_{i=K}^H \sum_{j=0}^N \gamma^{i*N+j} r_{i,j}$ is the discounted return in the MDP \mathcal{M} starting at the K th episode of the meta-trajectory \mathcal{D} , $r_{i,j}$ is the j th reward in episode i , and H is the length of the trial, or the *task-horizon*. This objective discounts rewards across episodes in \mathcal{D} . The exploration period is captured by K , also called the *shot*, which is the index of the first episode during the trial in which return counts towards the objective. $K = 0$ corresponds to no free exploration episodes.

The meta-RL objective defined by Equation 2.3 is evaluated in expectation over samples from the task distribution, $p(\mathcal{M})$. During meta-training, the task distribution $p(\mathcal{M})$ is usually accessed only via sampling. This results in a generalization problem in meta-testing, as testing tasks may not match the training tasks. This problem is made worse when the meta-testing task distribution does not share support with the training task distribution. Such out-of-distribution cases are often studied in meta-RL.

2.3 POMDP Formalization

The problem setting defining meta-RL, posed by Equation 2.3, can additionally be viewed as a special case of a partially observable Markov decision process (POMDP) (Duan *et al.*, 2016b; Humplik *et al.*, 2019). We specify the POMDP as a tuple of $\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \Omega, P, P_0, R, O, \gamma, H \rangle$. Here, Ω is the set of observations, O is the observation function, H is the task-horizon, and the rest are defined as in an MDP. To resolve ambiguity, when a function or variable depends on a particular MDP, we use a superscript to indicate the dependence, e.g., $R^{\mathcal{M}}$, for a reward function that depends on the MDP, \mathcal{M} . The POMDP states, actions, and transitions function similarly to an MDP. However, unlike in an MDP, the agent does not observe the current state, s_t , which is hidden from the agent. Instead, the agent observes an observation, $o_t \in \Omega$, with its probability distribution defined by $O(o_t|s_t, a_t) : \Omega \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_+$. The POMDP policy is then a function that maps a history of observations to a distribution over actions: $\pi(a|\tau_{:t}) : \mathcal{A} \times \Omega^t \rightarrow \mathbb{R}_+$.

Given a distribution of MDPs, $p(\mathcal{M})$, the POMDP corresponding to the meta-RL problem has a specific structure. In particular, it maintains the identity of an MDP sampled from this distribution, i.e., the reward function and the dynamics function, in its hidden state. Therefore, the state of the POMDP is the identity and state of the sampled MDP: $s_t = (\mathcal{M}, s_t^{\mathcal{M}})$. The initial state distribution of the POMDP is responsible for sampling an MDP, \mathcal{M} , and an initial MDP state, $s_0^{\mathcal{M}}$, given the MDP: $P_0(s_0) = p(\mathcal{M})P_0^{\mathcal{M}}(s_0^{\mathcal{M}})$. The reward function of the POMDP is the same as the reward function of the underlying MDP: $R(s_t, a_t) = R^{\mathcal{M}}(s_t^{\mathcal{M}}, a_t)$. The observation function deterministically returns the state of the underlying MDP, along with the previous action, and the reward: $o_t = (s_t^{\mathcal{M}}, a_{t-1}, r_{t-1})$. In meta-RL, the unknown MDP identity includes an unknown reward function. In order for the inner-loop to learn an optimal policy for the MDP, it requires information about the unknown reward function. Therefore, the observation includes the sampled reward, r_{t-1} , which is not always the case for observation functions for POMDPs considered outside of meta-RL. The dynamics function of the POMDP is that of the sampled MDP stored in the hidden state. Additionally, after N timesteps of the MDP, the dynamics

function of the POMDP must be constructed so that it transitions to initial states sampled from $P_0^{\mathcal{M}}$ of the underlying MDP. Formally,

$$s_{t+1} = \begin{cases} (\mathcal{M}, s_0^{\mathcal{M}} \sim P_0^{\mathcal{M}}) & \text{if } t \equiv 0 \pmod{N}, \\ (\mathcal{M}, s_{t+1}^{\mathcal{M}}) & \text{otherwise.} \end{cases}$$

A single episode in this particular type of POMDP thus consists of an entire trial, itself comprised of H episodes in the sampled MDP. When the task-horizon, H , is reached, the POMDP episode terminates. In the beginning of the next POMDP episode, a new POMDP state, including a new MDP, is sampled from the initial state distribution P_0 . Note that the transition function depends on the timestep, t , which can be stored in s , but is omitted from the state for brevity. Also, note that the history of observations in this POMDP, τ , is the concatenation of observations, $(s_t^{\mathcal{M}}, a_{t-1}, r_{t-1})$, across the H MDP episodes. Together, the sequence of these observations is equivalent to the dataset \mathcal{D} from Section 2. (This formalism can also be cast as a Contextual MDP, or CMDP, Hallak *et al.*, 2015, where the context is not shown to the agent.) This perspective leads to the following insights into the meta-RL problem.

First, from the theory of POMDPs, we know that the optimal policy is either history-dependent or dependent on a sufficient statistic, or the information state for the POMDP (Subramanian *et al.*, 2022). For the POMDP representing the meta-RL problem, it is therefore possible to use either representation. In the case that the inner-loop, f_θ , approximates a general function of history, then the inner-loop and the policy it produces, $\pi_\phi^{\mathcal{M}}$, can be viewed together as a single object, forming a history-dependent policy, $\pi_\theta(a|\tau)$. While such history-dependent policies do not take into account the specific structure of the meta-RL problem beyond that of a more general POMDP, they are sufficient for solving the meta-RL problem. Alternatively, the optimal policy may condition on a sufficient statistic that summarizes the history. In a POMDP, one such statistic is the posterior distribution over POMDP states, $b(s) : \mathcal{S} \rightarrow \mathbb{R}_+$ (Subramanian *et al.*, 2022), often referred to as the belief state. In the case that the inner-loop approximates such a belief state, the combined object can be seen as a policy dependent on approximations of a sufficient statistic, $\pi_\theta(a|\hat{b})$, where the inner-loop

computes $\hat{b} = f_\theta(\tau)$. The form of these sufficient statistics can be more specific than in the general POMDP, since the hidden state, s , has a specific form in meta-RL. In particular, one sufficient statistic for meta-RL, given the current observable MDP state, is the posterior distribution over tasks, $b = p(\mathcal{M} \mid \tau) = p(R^\mathcal{M}, P^\mathcal{M} \mid \tau)$, which we discuss further in Section 3.5. This dichotomy in POMDP methods, between history-dependent and belief-dependent, leads to two different meta-RL methods: black box methods, discussed in Section 3.2, and task inference methods, discussed in Section 3.3, respectively.

Second, this perspective enables us to draw connections to Bayesian RL (Duff and Barto, 2002; Ghadirzadeh *et al.*, 2021). In particular, Bayesian RL solves this POMDP by explicitly maintaining a posterior over tasks and updating it using Bayesian inference. The Bayesian framework provides a convenient method for incorporating prior knowledge and explicitly maintaining uncertainty. Using this framework a new MDP can be constructed, where the state includes the posterior over POMDP hidden states, or equivalently, the MDP state and a posterior over tasks. In this case, the resulting MDP is called a Bayes-adaptive Markov decision process (BAMDP). This construction allows learning a Markovian policy to solve the meta-RL problem and therefore explores optimally. BAMDPs and Bayes-optimal policies are discussed in Section 3.5. However, since Bayesian RL methods must explicitly model and update a distribution over MDPs, they are tractable only in simple domains without strong approximations. For example, even scalable Bayesian RL methods may be limited to discrete-space MDPs (Guez *et al.*, 2013). Instead of engineering approximations for the Bayesian posterior, meta-RL methods may learn to model these components as needed. For example, most meta-RL methods only require sample access to the prior instead of the prior being explicitly known. The meta-RL agent may learn to implicitly model the prior over tasks from samples.

Finally, while Meta-RL generally considers a distribution over MDPs, it is also possible to consider a distribution over POMDPs. In this case, each task is itself partially observable. This forms yet another type of POMDP, called a meta-POMDP (Akuzawa *et al.*, 2021). The meta-POMDP can be written as a POMDP where only a part of the hidden

state remains constant throughout a trial, and it is possible to adapt existing methods to accommodate this structure (Akuzawa *et al.*, 2021).

2.4 Example Algorithms

We now describe two canonical meta-RL algorithms that optimize the objective given by Equation 2.3: Model-Agnostic Meta-Learning (MAML), which uses meta-gradients (Finn *et al.*, 2017a), and Fast RL via Slow RL (RL²), which uses a history dependent policy (Duan *et al.*, 2016b; Wang *et al.*, 2016). Many meta-RL algorithms are based on concepts and techniques similar to those used in MAML and RL², making them excellent entry points to meta-RL.

MAML Many designs of the inner-loop algorithm f_θ build on existing RL algorithms and use meta-learning to improve them. MAML (Finn *et al.*, 2017a) is an influential design following this pattern. Its inner-loop is a policy gradient algorithm whose initial parameters are the meta-parameters $\phi_0 = \theta$. The key insight is that such an inner-loop is a differentiable function of the initial parameters, and therefore the initialization can be optimized with gradient descent to be a good starting point for learning on tasks from the task distribution. When adapting to a new task, the inner-loop MAML collects data using the initial policy and computes an updated set of parameters by applying a policy gradient step for a task $\mathcal{M}^i \sim p(\mathcal{M})$:

$$\phi_1^i = f_\theta(\mathcal{D}_0^i) = \phi_0 + \alpha \nabla_{\phi_0} \hat{J}(\mathcal{D}_0^i, \pi_{\phi_0}),$$

where $\hat{J}(\mathcal{D}_0^i, \pi_{\phi_0})$ is an estimate of the returns of π_{ϕ_0} in task \mathcal{M}^i computed on data \mathcal{D}_0^i collected using π_{ϕ_0} . Note that $\phi_0 = \theta$ is a learnable meta-parameter, which is not specific to any task. In contrast, ϕ_1^i are the adapted policy parameters that are specific to the task \mathcal{M}^i . Since ϕ_0 affects the initial policy, it not only determines initial rewards, but also determines the data collected to produce ϕ_1^i .

To learn θ in the outer-loop, it must be updated based on its effect on the initial policy and its effect on ϕ_1^i . To differentiate through the adaptation process, MAML collects a second batch of data \mathcal{D}_1^i using

the policy $\pi_{\phi_1^i}$ and computes the gradient of the returns of the updated policy with respect to the initial parameters across the tasks given by

$$\frac{1}{M} \sum_{i=0}^M \nabla_{\phi_0} \hat{J}(\mathcal{D}_1^i, \pi_{\phi_1^i}),$$

where M is the number of sampled tasks, $\pi_{\phi_1^i}$ is the policy for task \mathcal{M}^i updated once by the inner-loop, and $\nabla_{\phi_0} \hat{J}(\mathcal{D}_1^i, \pi_{\phi_1^i})$ is the gradient of the returns of the updated policy computed w.r.t. the initial policy parameters. Such a gradient through an RL inner-loop is often referred to as a *meta-gradient*. Descending the meta-gradient corresponds to optimizing the outer-loop objective given by Equation 2.3. The version of MAML described here considers only a single update in the inner-loop but for harder problems, the inner-loop can compute multiple updates. The additional updates do not change the meta-gradient computation. The algorithm is illustrated in Algorithm 1 and Figure 2.3.

Algorithm 1 MAML for Reinforcement Learning

- 1: Initialize meta-parameters, θ , which here are the initial policy parameters, ϕ_0 .
 - 2: **while** not done **do**
 - 3: Sample M tasks, $\mathcal{M} \sim p(\mathcal{M})$
 - 4: **for** each task index, i **do**
 - 5: Collect data \mathcal{D}_0^i using the initial policy π_{ϕ_0} .
 - 6: Adapt policy parameters using a policy gradient step: $\phi_1^i \leftarrow \phi_0 + \alpha \nabla_{\phi_0} \hat{J}(\mathcal{D}_0^i, \pi_{\phi_0})$
 - 7: Collect data \mathcal{D}_1^i using the updated policy $\pi_{\phi_1^i}$.
 - 8: **end for**
 - 9: Update ϕ_0 using the meta-gradient: $\phi_0 \leftarrow \phi_0 + \beta \nabla_{\phi_0} \frac{1}{M} \sum_i \hat{J}(\mathcal{D}_1^i, \pi_{\phi_1^i})$.
 - 10: **end while**
-

Typically in MAML, the shot K and task-horizon H are chosen such that the outer-loop objective only considers the returns of the last policy produced by the inner-loop. In the simplest case, this would mean setting $K = 1$ and $H = 2$, i.e., using one episode for computing the inner-loop update and another for computing the outer-loop objective. To

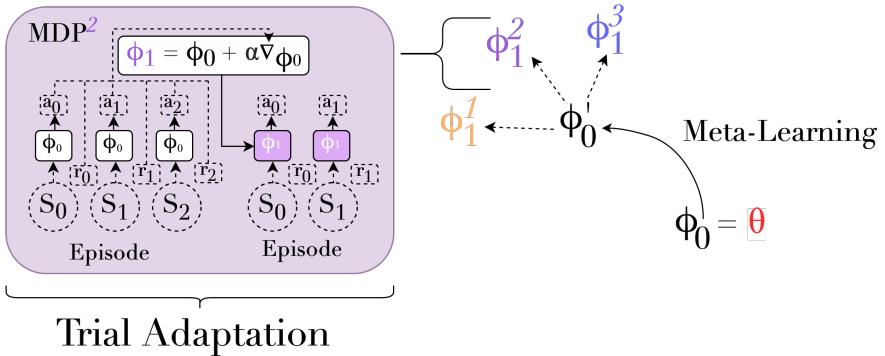


Figure 2.3: MAML in the problem setting (left) and conceptually (right). The meta-parameters θ are the initial parameters of the inner-loop policies ϕ_0 . The inner-loop computes new parameters ϕ_1^i adapted to task i using one step of a policy gradient algorithm. The outer-loop updates the meta-parameters, from ϕ_0 to ϕ_0' , to optimize the performance of the policies after adaptation.

accommodate multiple updates in the inner-loop and sampling multiple episodes with each policy for variance reduction, higher values of K and H are often used.

RL² Another popular approach to meta-RL is to represent the inner-loop as a policy that is dependent on *the entire history* of its interaction with the environment and train it on tasks from the task distribution end-to-end with RL. As discussed in Section 2.3, when f_θ is represented as a sequence model, f_θ and $\pi_{f_\theta(\mathcal{D})}(a|s)$ can be viewed together as a single history-dependent policy, $\pi_\theta(a|s, \mathcal{D})$. This approach is the same as applying a history-dependent policy, as used in the more general POMDP setting, to meta-RL as a special case. This history includes all of the states, actions, and rewards the policy encounters during a trial. This results in learning an adaptive policy that changes its behavior as it gathers more information about the environment. A similar idea has been explored for supervised learning, for example by Hochreiter *et al.* (2001). For meta-RL specifically Duan *et al.* (2016b) and Wang *et al.* (2016) propose a method called RL², where the history dependent policy is represented as a recurrent neural network (RNN). Using ϕ to represent the RNN hidden state, we can write the recurrent policy

as $\pi_\theta(a|s, \phi)$. While in this example we focus on RNNs, any history dependent policy could be used.

To optimize the meta-RL objective from Equation 2.3, RL² treats the trial as a single continuous sequence, during which the RNN hidden state is not reset, even if the trial spans multiple episodes in the underlying MDP. The meta-parameters θ are the parameters of the RNN and other neural networks used in processing the inputs and outputs of f_θ . The task parameters ϕ are the ephemeral hidden states of the RNN, which may change after every timestep. The operation of the algorithm is illustrated in Algorithm 2 and Figure 2.4.

Algorithm 2 RL² for Meta-Reinforcement Learning

- 1: Initialize meta-parameters θ (RNN and other neural network parameters)
 - 2: **while** not done **do**
 - 3: Sample M tasks, $\mathcal{M} \sim p(\mathcal{M})$
 - 4: **for** each task index, i **do**
 - 5: Initialize RNN hidden state ϕ_0^i
 - 6: Run a continuous trial consisting of multiple episodes
 - 7: **for** each timestep t in the trial **do**
 - 8: Observe current state s_t , previous action a_{t-1} , and previous reward r_{t-1}
 - 9: Update RNN hidden state with input $[s_t, a_{t-1}, r_{t-1}]$:
- $$\phi_t^i \leftarrow f_\theta(s_t, a_{t-1}, r_{t-1}, \phi_{t-1}^i)$$
- 10: Sample action $a_t \sim \pi_\theta(\cdot|s_t, \phi_t^i)$
 - 11: Execute action a_t and receive reward r_t
 - 12: **end for**
 - 13: **end for**
 - 14: Update meta-parameters by optimizing Equation 2.3: $\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{M} \sum_i G(\mathcal{D}^i)$.
 - 15: **end while**
-

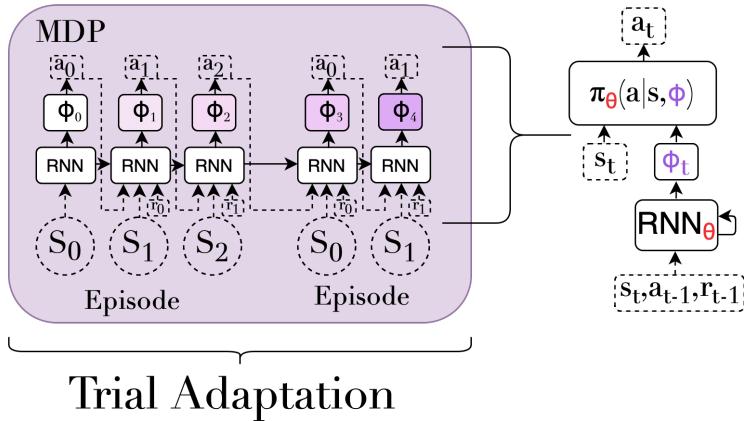


Figure 2.4: RL^2 in the problem setting (left) and conceptually (right). The inner-loop algorithm is implemented by an RNN parameterized by the meta-parameters θ . The RNN takes as input the states, actions, and rewards from the environment. The RNN hidden state ϕ_t defines the task parameters at each timestep, which are passed as input to the MLP policy. The hidden state is not reset during a trial and instead carries over across episode boundaries. The outer-loop is a standard RL algorithm.

These two distinct approaches to meta-RL, MAML and RL^2 , each bring their unique advantages and disadvantages. On one hand, the generality of MAML’s policy gradient algorithm in its inner-loop allows it, under specific conditions, to learn a policy starting from its initial state for *any* given task, including those outside the task distribution. On the other hand, RL^2 directly approximates the optimal policy of the meta-RL objective, given by Equation 2.3. This policy, known as Bayes-optimal, is the best policy for the task distribution and is further discussed in Section 3.5. Bayes-optimal policies always choose actions that maximize the expected return under uncertainty about the MDP identity, whereas the policy-gradient dependent MAML can only update when full episodes have been collected in the inner-loop. However, a drawback of RL^2 is that it faces a challenging generalization problem when tested on tasks outside the task distribution. The end-to-end RL training on a narrow task distribution may not result in a policy that generalizes well outside the task distribution.

2.5 Problem Categories

While the given problem setting applies to all of meta-RL, distinct clusters in the literature have emerged based on two dimensions: whether the task-horizon H is short (a few episodes) or long (hundreds of episodes or more), and whether the task distribution $p(\mathcal{M})$ contains multiple tasks or just one. This creates four clusters of problems, of which three yield practical algorithms, as shown in Table 2.1. We illustrate how these categories differ in Figure 2.5.

Table 2.1: Example methods for the three categories of meta-RL problems. The categorization is based on whether they consider multi-task task distributions or a single task and few or many shots.

	Multi-task	Single-task
Few-Shot	RL ² (Duan <i>et al.</i> , 2016b; Wang <i>et al.</i> , 2016), MAML (Finn <i>et al.</i> , 2017a)	-
	LPG (Oh <i>et al.</i> , 2020), MetaGenRL (Kirsch <i>et al.</i> , 2019)	STAC (Zahavy <i>et al.</i> , 2020), FRODO (Xu <i>et al.</i> , 2020)
Many-Shot		

The first cluster in Table 2.1 is the *few-shot multi-task setting*. In this setting, an agent must quickly – within just a few episodes – adapt to a new MDP sampled from the task distribution it was trained on. This requirement captures the central idea in meta-RL that we want to use the task distribution to train agents that are capable of learning new tasks from that distribution using as few environment interactions as possible. The *shots* in the name of the setting echo the shots in *few-shot classification* (Vinyals *et al.*, 2016; Snell *et al.*, 2017; Finn *et al.*, 2017a), where a model is trained to recognize new classes given only a few samples from each. In meta-RL, the number of exploration episodes K is more or less analogous to the shots in classification. This setting is discussed in detail in Section 3.

While few-shot adaptation directly tackles the motivating question for meta-RL, sometimes the agent faces an adaptation problem so difficult that it is unrealistic to hope for success within a small number of episodes. This might happen, for example, when adapting to tasks that do not have support in the task distribution. In such cases, the

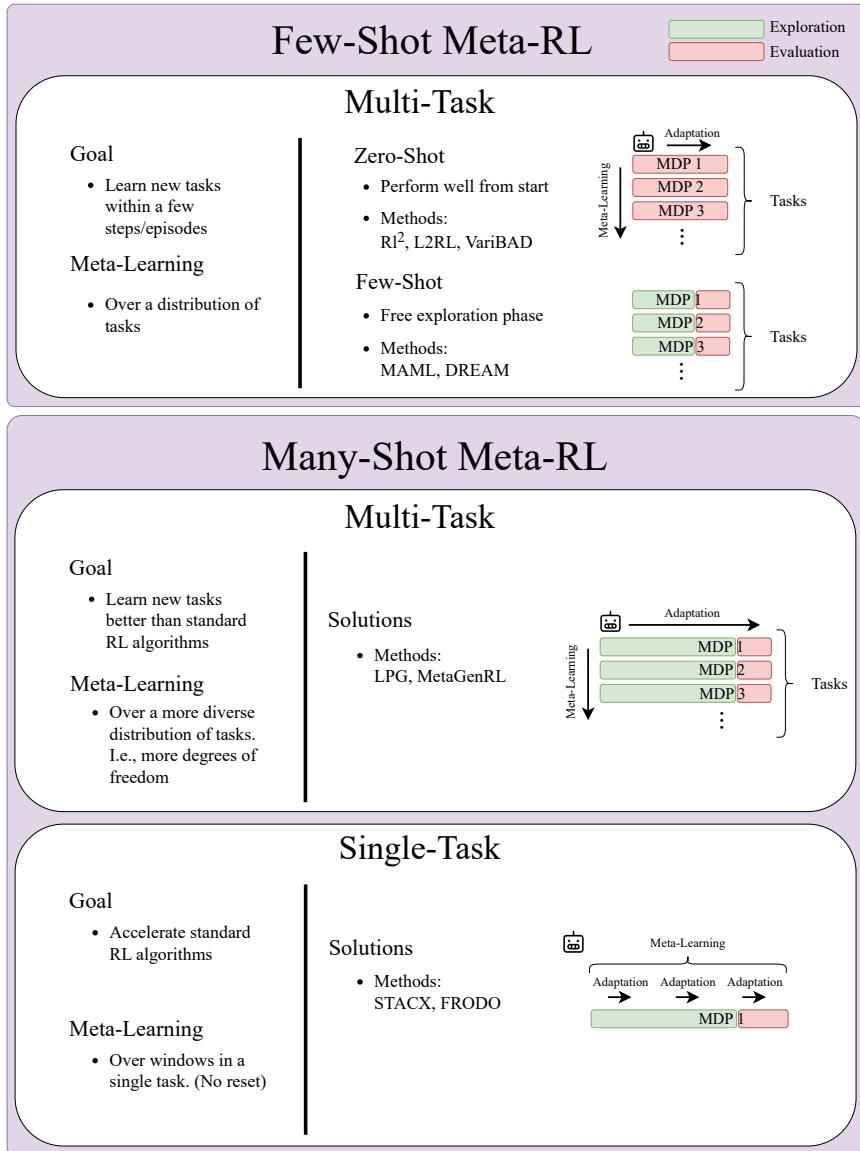


Figure 2.5: Illustration of the different meta-RL settings considered in this survey. Each setting is summarized in terms of its motivating goal, task distribution, and the defining features of its algorithms. The small embedded figures on the right illustrate the adaptation over task distributions in each setting. The tasks are shown as bars depicting trials with the shot and task-horizon depicted by the lengths of the green and red bars.

inner-loop may require thousands of episodes or more to produce a good policy for a new task, but we would still like to use meta-learning to make the inner-loop as data-efficient as possible. We discuss this *many-shot multi-task* setting in Section 4.

Meta-RL is mostly concerned with the multi-task setting, where the inner-loop can exploit similarities between the tasks to learn a more data-efficient adaptation procedure. In contrast, in standard RL, agents are often trained to tackle a single complex task over many optimization steps. Since this is often highly data-inefficient, researchers have investigated whether meta-learning methods can improve efficiency even without access to a distribution of related tasks. In such cases, the efficiency gains must come from transfer within the single lifetime of the agent or from adaptation to the local training conditions during training. Methods for this *many-shot single-task* setting tend to resemble those in the many-shot multi-task setting and are therefore also discussed in Section 4.

The fourth cluster is the *few-shot single-task* setting, where a hypothetical meta-learner would accelerate learning on a single task within a few episodes without transfer from other related tasks. The short lifetime of the agent is unlikely to leave enough time for the inner-loop to learn a data-efficient adaptation procedure and produce a policy using it. Therefore, there is to our knowledge no research targeting this setting.

2.6 Related Work

Related Fields The objective of this survey is to introduce, summarize, and highlight research in the field of meta-reinforcement learning. Several existing surveys (Thrun and Pratt, 1998; Hospedales *et al.*, 2020; Wang *et al.*, 2020b; Parker-Holder *et al.*, 2022; Kirk *et al.*, 2023) include some meta-RL methods, but differ in a few key ways. Thrun and Pratt (1998) and Hospedales *et al.* (2020) both include meta-RL methods, but focus on meta-learning more broadly, including meta-supervised learning. Wang *et al.* (2020b) survey methods for few-shot learning, but meta-RL is broader than few-shot learning, since meta-RL also includes many-shot problems, and meta-RL additionally requires a

task-distribution for learning in the few-shot setting. Parker-Holder *et al.* (2022) survey methods for AutoRL, but also include non-learned algorithms for adapting to new MDPs. Kirk *et al.* (2023) survey methods for generalization in RL, which includes meta-RL methods, but also includes both non-learned algorithms and problems that do not require adaptation, i.e., problems where a single policy can perform optimally without specialization to the given task. Of all the related machine learning fields, meta-supervised learning (meta-SL) and multi-task RL, are the most closely related.

Meta-Supervised Learning There are many points of similarity between meta-RL and meta-SL research. Like meta-RL, meta-SL considers a distribution of tasks. However, whereas the tasks in meta-RL are defined by MDPs, the tasks in meta-SL are defined by fixed datasets. Some algorithms have been proposed as methods for both meta-SL and meta-RL (Finn *et al.*, 2017a; Mishra *et al.*, 2018). Moreover, many algorithmic choices have analogues in both settings. For example, in the few-shot setting, the use of attention over prior states in meta-RL (Ritter *et al.*, 2018a; Fortunato *et al.*, 2019; Team *et al.*, 2023), has analogues in the kernel-based methods of supervised meta-SL (Santoro *et al.*, 2016; Vinyals *et al.*, 2016; Sung *et al.*, 2018). These methods are covered later in Section 3.3. Additionally, in the many-shot setting, meta-learning optimizers and objective functions has been covered in both meta-RL (Houthooft *et al.*, 2018; Kirsch *et al.*, 2019; Oh *et al.*, 2020; Bechtle *et al.*, 2021; Lu *et al.*, 2022a; Lan *et al.*, 2023) and meta-SL (Andrychowicz *et al.*, 2016; Li and Malik, 2017; Bechtle *et al.*, 2021). These methods are covered in Section 4.3. Finally, imitation learning is considered a closely related pursuit to RL, since both consider sequential decision making problems. Therefore, we discuss meta-supervised learning for imitation learning in Section 3.6.

In addition to similarities in the problem settings and methods, meta-RL and meta-SL have some key differences. In particular, meta-SL generally considers fixed datasets while meta-RL does not. In meta-RL, the adapted policy is responsible for collecting more data, which is fed back into the dataset for adaptation. This introduces a problem of data collection for adaptation to the given task. Moreover, since the meta-RL

agent must adapt quickly, this requires that the agent also explores quickly. Fast exploration and the resulting challenges are unique to meta-RL and are a key theme discussed at length in Section 3.

Multi-Task RL Multi-task RL, like meta-RL, considers a distribution of MDPs. However, whereas a meta-RL agent must identify the MDP that it encounters, in multi-task RL, the agent has access to a ground-truth representation of the task. While some solutions from multi-task RL, such as PopArt (Hessel *et al.*, 2019), are immediately applicable to meta-RL (Grigsby *et al.*, 2024), differences in evaluation also prevent some methods from being applicable. For example, multi-task RL is often evaluated over a finite set of discrete tasks, whereas meta-RL agents often encounter new tasks at test time (Yu *et al.*, 2020a; Teh *et al.*, 2017). This allows multi-task methods to train separate networks for each task (Teh *et al.*, 2017), whereas such approaches are not immediately applicable in meta-RL.

Generally, multi-task RL can be seen as an easier version of meta-RL, in the sense that the MDP representation is known, so there is no need to learn it from data, nor to explore to get data for adaptation. In fact, an entire category of meta-RL methods tries to explicitly infer the MDP identity in order to reduce the meta-RL problem to the easier multi-task RL problem, which is discussed in Section 3.3. However, there can also be cases in which meta-RL is possible and multi-task RL is not. Since generalization to different tasks in multi-task RL occurs without conditioning on data at test time, it is always zero-shot generalization. Such generalization may not be possible, in practice, if the task representation is not sufficiently informative. For example, if the task is represented as a one-hot categorical vector, then it may be impossible to generalize to categories not seen during training. However, while multi-task RL would fail, meta-learning may still be viable in this case, if a more informative representation of the task can be inferred from data, or if additional learning at test time can compensate for sparsity in the training distribution.

3

Few-shot Meta-RL

In this section, we discuss few-shot adaptation, where the agent meta-learns across multiple tasks and at meta-test time must quickly adapt to a new, but related task in a few timesteps or episodes.

As a concrete example, recall the robot chef learning to cook in home kitchens. Training a new policy to cook in each user’s home using reinforcement learning from scratch would require many samples in each kitchen, which can be wasteful as general cooking knowledge (e.g., how to use a stove) transfers across kitchens. Wasting many data samples in the homes of customers who purchased a robot may be unacceptable, particularly if every action the robot takes risks damaging the kitchen. Meta-RL can automatically learn a procedure, from data, for adapting to the differences that arise in new kitchens (e.g., the location of the cutlery). During meta-training, the robot may train in many different kitchens in simulation or a setting with human oversight and safety precautions. Then, during meta-testing, the robot is sold to a customer and deployed in a new kitchen, where it must quickly learn to cook in it. However, training such an agent with meta-RL involves unique challenges and design choices.

In particular, here we summarize the literature in terms of the inner-loop, the exploration, the supervision, and whether the RL algorithm is

model-based or model-free. We categorize by the type of inner-loop, since the majority of research considers different inner-loop parameterizations. We consider exploration, since learning exploration is unique to meta-RL compared to meta-SL, as discussed in Section 2.6. We classify closely related problem settings, which utilize imitation learning, and some other forms of supervision, by the supervision available. Finally, we survey model-based approaches to meta-RL, since they present different trade-offs to model-free methods. However, we only discuss model-based methods briefly, as most meta-learning literature concerns the model-free setting.

Meta-parameterization In this section, we first discuss three common categories of methods in the multi-task few-shot setting. Methods in these categories are further classified in Table 3.1. Recall that meta-RL itself learns a learning algorithm f_θ . This places unique demands on f_θ and suggests particular representations for this function. We call this design choice the *meta-parameterization*, with the most common ones being the following:

- **Parameterized policy gradient** methods build the structure of existing policy gradient algorithms into f_θ . We review these in Section 3.1.
- **Black box** methods impose little to no structure on f_θ . We review these in Section 3.2.
- **Task inference** methods structure f_θ to explicitly infer the unknown task. We review these in Section 3.3.

While each of these categories represents a discrete cluster of research, other authors cluster the research differently and may use different names to refer to these clusters. For example, parameterized policy gradient methods are sometimes referred to as *gradient-based* methods (Finn *et al.*, 2017a; Rakelly *et al.*, 2019), or are referred to as part of a larger category of methods in meta-SL called *optimization-based* methods (Hospedales *et al.*, 2020). Additionally, black box methods

Table 3.1: Representative examples of few-shot meta-RL research categorized by method. The majority of methods fall into one of three clusters: Parameterized policy gradient, black box, or task inference. These categories determine how the inner-loop is parameterized. Within each cluster, we further categorize the methods. Explanations of these methods can be found in Sections 3.1, 3.2, and 3.3, respectively.

Parameterized Policy Gradients	
MAML-like	Finn <i>et al.</i> (2017a), Zintgraf <i>et al.</i> (2019), Park and Oliva (2019), and Raghu <i>et al.</i> (2020)
Adapt Policy Distribution	Yoon <i>et al.</i> (2018), Gupta <i>et al.</i> (2018b), Wang <i>et al.</i> (2020c), Zou and Lu (2020), and Ghadirzadeh <i>et al.</i> (2021)
Meta-gradient estimation	Al-Shedivat <i>et al.</i> (2018a), Stadie <i>et al.</i> (2018), Foerster <i>et al.</i> (2018a), Fallah <i>et al.</i> (2020), and Tang (2022)
Alternative outer-loop algorithms	Sung <i>et al.</i> (2017), Mendonca <i>et al.</i> (2019), and Song <i>et al.</i> (2020a)
Black Box	
RL2-Like	Heess <i>et al.</i> (2015), Duan <i>et al.</i> (2016b), and Wang <i>et al.</i> (2016)
Adapt Policy Using Context Vector	Duan <i>et al.</i> (2016b), Wang <i>et al.</i> (2016), Zintgraf <i>et al.</i> (2019), Humplik <i>et al.</i> (2019), Zintgraf <i>et al.</i> (2020), Fakoor <i>et al.</i> (2020), and Liu <i>et al.</i> (2021)
Inner-Loop with Hebbian Learning	Miconi <i>et al.</i> (2018), Miconi <i>et al.</i> (2019), Najarro and Risi (2020), Chalvidal <i>et al.</i> (2022), and Rohani <i>et al.</i> (2022)
Inner-Loop with Attention	Oh <i>et al.</i> (2016), Ritter <i>et al.</i> (2018b), Mishra <i>et al.</i> (2018), Fortunato <i>et al.</i> (2019), Ritter <i>et al.</i> (2021), Melo (2022), Xu <i>et al.</i> (2022), Team <i>et al.</i> (2023), and Elawady <i>et al.</i> (2024)
Task Inference	
Multi-task pre-training	Humplik <i>et al.</i> (2019), Kamienny <i>et al.</i> (2020), Raileanu <i>et al.</i> (2020), Peng <i>et al.</i> (2021), and Liu <i>et al.</i> (2021)
Without privileged information	Guo <i>et al.</i> (2018), Rakelly <i>et al.</i> (2019), Zintgraf <i>et al.</i> (2020), Fu <i>et al.</i> (2021), Zhang <i>et al.</i> (2021), and Luo <i>et al.</i> (2022)
Permutation invariance	Rakelly <i>et al.</i> (2019), Raileanu <i>et al.</i> (2020), Korshunova <i>et al.</i> (2020), Imagawa <i>et al.</i> (2022), and Beck <i>et al.</i> (2024)
Adapt Policy Using Hypernetworks	Peng <i>et al.</i> (2021), Beck <i>et al.</i> (2022), and Beck <i>et al.</i> (2023)

and task-inference methods are sometimes referred to collectively as *context-based* methods (Rakelly *et al.*, 2019).

Along with the parameterization of the inner-loop, related design choices include the representation of the base policy and the choice of the outer-loop algorithm. Each method must additionally specify which base policy parameters are adapted by the inner-loop, and which

are meta-parameters. We call this design choice the adapted policy parameters. For each of the three types of meta-parameterizations in this section, we discuss the inner-loop, the outer-loop, and the adapted policy parameters.

Exploration While all these methods are distinct, they also share some challenges. One such challenge is that of *exploration*, the process of collecting data for adaptation. In few-shot learning, exploration determines how an agent takes actions during its few shots. Subsequently, an agent must decide how to adapt the base policy using this collected data. For the adaptation to be sample efficient, the exploration must be efficient as well. Specifically, the exploration must target differences in the task distribution (e.g., different locations of cutlery). In Section 3.4, we discuss the process of exploration, along with ways to add structure to support it. While all few-shot methods must learn to explore an unknown MDP, there is an especially tight relationship between exploration methods and task inference methods. In general, exploration may be used to enable better task inference, and conversely, task inference may be used to enable better exploration. One particular way in which task inference may be used to enable better exploration is by quantifying uncertainty about the task, and then choosing actions based on that quantification. This method may be used in order to learn optimal exploration. We discuss this in Section 3.5.

Supervision Meta-RL methods also differ in the assumptions that they make about the available *supervision*. In the standard meta-RL problem setting, rewards are available during both meta-training and meta-testing. However, providing rewards in each phase presents challenges. For example, it may be difficult to manually design an informative task distribution for meta-training, and it may be impractical to measure rewards with expensive sensors during deployment for meta-testing. In this case, unique methods must be used, such as automatically designing rewards for the outer-loop, or creating an inner-loop that does not need to condition on rewards. Alternatively, supervision that is more informative than rewards can be provided. We review such settings, challenges, and methods in Section 3.6.

Model-Based Meta-RL Some meta-RL methods explicitly learn a model of the MDP dynamics and reward function. Such methods are called *model-based* methods, in contrast to *model-free* methods that do not explicitly learn a model of the environment. Model-based methods confer advantages such as sample-efficient and off-policy meta-training. Moreover, using an off-the-shelf planning algorithm with the learned model can be easier for some task distributions than learning a complicated policy directly. However, model-based methods generally require the implementation of additional components and can have lower asymptotic performance. We discuss these trade-offs and the applications of model-based RL to meta-RL in Section 3.7. We keep discussion in this section brief, since most meta-RL literature considers mode-free RL, the relevant trade-offs are similar in meta-RL and RL at large, and most model-based meta-RL methods have an analogous model-free method that will have already been discussed by that point.

Theory of Meta-RL Finally, we consider theoretical research on meta-RL. While meta-RL is a relatively new area of research, several studies have found interesting theoretical challenges in it. These insights relate well to the empirical findings, which constitute the majority of the research in meta-RL. In Section 3.8, we survey important theoretical works in meta-RL. We believe there is a lot more to explore in the theory of meta-RL and therefore hope this survey can motivate future research.

3.1 Parameterized Policy Gradient Methods

Meta-RL learns a learning algorithm f_θ , the inner-loop. This inner-loop is learned to maximize the meta-RL objective given by Equation 2.3, over samples from a task distribution, $p(\mathcal{M})$. However, during training, we generally only assume access to samples from this distribution. These samples may be limited, and the distribution over which we want to generalize may be broad. Moreover, the distribution on which the meta-RL is evaluated for meta-testing may differ from the distribution on which the meta-RL agents is meta-trained. Each of these motivates the need for learning to take place in the inner-loop algorithm f_θ . This section discusses methods for building such structure into the inner-loop itself.

We call the parameterization of f_θ the *meta-parameterization*. In this section, we discuss one way of parameterizing the inner-loop that builds in the structure of existing standard RL algorithms. *Parameterized policy gradients (PPG)* are a common class of methods which parameterize the learning algorithm f_θ as a policy gradient algorithm. These algorithms generally have an inner-loop of the form

$$\phi_{j+1} = f_\theta(\phi_j, \mathcal{D}_j) = \phi_j + \alpha_\theta \nabla_{\phi_j} \hat{J}_\theta(\mathcal{D}_j, \pi_{\phi_j}),$$

where $\hat{J}_\theta(\mathcal{D}_j, \pi_{\phi_j})$ is an estimate of the returns of the policy π_{ϕ_j} . For example, recall the MAML algorithm discussed in Section 2.4. In MAML, $\theta = \phi_0$, and so the initialization is the meta-learned component. It is also possible to add additional pre-defined components to the inner-loop, such as sparsity-inducing regularization (Lou *et al.*, 2021). In general, whatever structure is not predefined, is a parameter in θ that is meta-learned. In addition to initialization, the meta-learned structure can include components such as hyper-parameters (Li *et al.*, 2017). Some methods also meta-learn a preconditioning matrix to approximate the curvature of the objective, inspired by second-order optimization methods. These methods generally have the form $\phi_{j+1} = \phi_j + \alpha_\theta M_\theta \nabla_{\phi_j} \hat{J}_\theta(\mathcal{D}_j, \pi_{\phi_j})$ (Park and Oliva, 2019; Flennerhag *et al.*, 2020). While a value based-method could be used to parameterize the inner-loop instead of a policy gradient (Zou *et al.*, 2021), value based-methods generally require many more steps to propagate reward information (Mitchell *et al.*, 2020), and so are typically reserved for the many-shot setting, discussed in Section 4. In this section, we focus on PPG methods. We begin by discussing different parameters of the base policy that the inner-loop may adapt. Then, we discuss options for outer-loop algorithms and optimization. We conclude with a discussion of the trade-offs associated with PPG methods.

Adapted policy parameters PPG algorithms commonly meta-learn an initialization, and then adapt that initialization in the inner-loop. Instead of adapting a single initialization, several PPG methods learn a full distribution over initial policy parameters, $p(\phi_0)$ (Yoon *et al.*, 2018; Gupta *et al.*, 2018b; Wang *et al.*, 2020c; Zou and Lu, 2020; Ghadirzadeh *et al.*, 2021). This distribution allows for modeling uncertainty over

policies. The distribution over initial parameters can be represented with a finite number of discrete particles (Yoon *et al.*, 2018), or with a Gaussian approximation fit via variational inference (Gupta *et al.*, 2018b; Ghadirzadeh *et al.*, 2021). Moreover, the distribution itself can be updated in the inner-loop, to obtain a posterior over (a subset of) model parameters (Yoon *et al.*, 2018; Gupta *et al.*, 2018b). The updated distribution may be useful both for modeling uncertainty (Yoon *et al.*, 2018), and for temporally extended exploration, if policy parameters are resampled periodically (Gupta *et al.*, 2018b).

Alternately, some PPG methods adapt far fewer parameters in the inner-loop. Instead of adapting all policy parameters, they adapt a subset (Zintgraf *et al.*, 2019; Raghu *et al.*, 2020). For example, one method adapts only the weights and biases of the last layer of the policy (Raghu *et al.*, 2020), while leaving the rest of the parameters constant throughout the inner-loop. Another method adapts only a vector, called the *context vector*, on which the policy is conditioned (Zintgraf *et al.*, 2019). In this case, the input to the policy itself parameterizes the range of possible behavior. We write the policy as $\pi_\theta(a|s, \phi)$, where ϕ is the adapted context vector. The weights and biases of the policy, as well as the initial context vector, constitute the meta-parameters that are constant in the inner-loop. We visualize the use of a context vector in Figure 3.1.

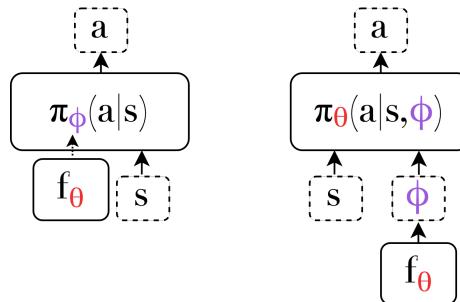


Figure 3.1: Illustration of meta-RL without (left) and with (right) a context vector. The context vector can be updated with backpropagation in a PPG method or by a neural network in a black box method. When using a context vector, some of the policy parameters are not adapted by the inner-loop, and are instead meta-parameters set by the outer-loop.

Consider the algorithm Context Adaptation via Meta-learning (CAVIA) (Zintgraf *et al.*, 2019). CAVIA is similar to MAML but it only adapts a context vector, which is initialized to the zero vector:

$$\begin{aligned}\phi_0 &= \mathbf{0}, \\ \phi_{j+1} &= \phi_j + \alpha_\theta \nabla_{\phi_j} \hat{J}_\theta(\mathcal{D}_j, \pi_\theta(\cdot | \phi_j)).\end{aligned}$$

While the update to ϕ is the same as in MAML, here, ϕ represents a vector passed as an input to the policy network, while all the weights and biases of the network remain fixed throughout adaptation. One benefit of adapting a subset of parameters in the inner-loop is that it may mitigate overfitting in the inner-loop, for task distributions where only a small amount of adaption is needed (Zintgraf *et al.*, 2019).

Meta-gradient estimation in outer-loop optimization Effective learning in the outer-loop of PPG algorithms requires access to estimates of *meta-gradients*, i.e., gradients of the outer-loop objective with respect to the meta-parameters. Most commonly, the meta-gradients are computed by directly optimizing Equation 2.3 with a policy gradient algorithm. However, naively applying a policy gradient method in the outer-loop can lead to a suboptimal bias-variance trade-off. Significant research has considered how to improve this trade-off when estimating meta-gradients for PPG methods (Stadie *et al.*, 2018; Foerster *et al.*, 2018a; Rothfuss *et al.*, 2019; Mao *et al.*, 2019; Liu *et al.*, 2019; Fallah *et al.*, 2020; Vuorio *et al.*, 2021; Tang, 2022; Fallah *et al.*, 2021; Tang *et al.*, 2021; Tack *et al.*, 2022; Liu *et al.*, 2022a). Next, we discuss the bias-variance trade-offs in meta-gradient estimation for PPG methods and the algorithms arising from choosing different points in the trade-off.

The unbiased gradient of the meta-RL objective given by Equation 2.3 with respect to the meta-parameters θ can be computed as follows. Since the distribution of tasks does not depend on the meta-parameters, we can estimate the meta-gradient with respect to each task separately and then combine. The expression for the meta-gradient for a single task is given by

$$\nabla_\theta \mathbb{E}_{\mathcal{D}} \left[\sum_{k=K}^H G(\mathcal{D}_k) \middle| \pi_{f_\theta} \right] = \nabla_\theta \sum_{k=K}^H \int G(\mathcal{D}_k) \prod_{i=0}^k p(\mathcal{D}_i; \phi_i) d\mathcal{D}_i.$$

Here, we take into account that the returns from previous trajectories are not influenced by the policies of future trajectories. Therefore, we restricted the product of trajectory probabilities to range only from $i = 0$ to k . Since the probabilities of the trajectories depend on θ , we use the log-derivative trick to get the gradient of the product

$$= \sum_{k=K}^H \int G(\mathcal{D}_k) \prod_{i=0}^k p(\mathcal{D}_i; \phi_i) \nabla_\theta \log \prod_{i=0}^k p(\mathcal{D}_i; \phi_i) d\mathcal{D}_i.$$

Finally, we use the chain rule to get the meta-gradient:

$$= \sum_{k=K}^H \int G(\mathcal{D}_k) \prod_{i=0}^k p(\mathcal{D}_i; \phi_i) \sum_{j=0}^k \nabla_\theta \phi_j \nabla_{\phi_j} \log p(\mathcal{D}_j; \phi_j) d\mathcal{D}_i$$

Naively applying a policy gradient algorithm to the returns $\sum_{k=K}^H G(\mathcal{D}_k)$ does not compute this meta-gradient. To see why, we reorganize the terms to get

$$= \sum_{k=K}^H \mathbb{E}_{\mathcal{D}} \left[G(\mathcal{D}_k) \left(\nabla_\theta \phi_k \nabla_{\phi_k} \log p(\mathcal{D}; \phi_k) + \sum_{j=0}^{k-1} \nabla_\theta \phi_j \nabla_{\phi_j} \log p(\mathcal{D}_j; \phi_j) \right) \right],$$

where the first gradient term is the regular policy gradient on the k th trajectory w.r.t. the meta-parameters, and the gradient terms in the sum are the gradients of the policies on the j th trajectories for $0 \leq j \leq k-1$ w.r.t. the meta-parameters. These latter terms, sometimes called *sampling-correction* terms, are often omitted in practice, yielding biased meta-gradient estimates (Al-Shedivat *et al.*, 2018a; Stadie *et al.*, 2018).

Recall that PPG methods produce a sequence of policies, and generally optimize Equation 2.3 with $K = H - n$, where n is the number of episodes collected by the final policy. Actions taken by any one policy affect the data seen by the next, and thus these actions affect the expected return of the later policies. The sampling-correction terms account for this dependence. Ignoring them amounts to ignoring actions from earlier policies when training the inner-loop to optimize the meta-RL objective and thus introduces bias to the meta-gradient estimation (Al-Shedivat *et al.*, 2018a). This bias can sometimes be detrimental to the meta-learning performance but the high variance of the unbiased estimator

often dominates (Stadie *et al.*, 2018; Fallah *et al.*, 2020; Vuorio *et al.*, 2021).

Significant research has considered a meta-gradient estimator derived originally for computing higher-order meta-gradients (Foerster *et al.*, 2018a; Rothfuss *et al.*, 2019; Farquhar *et al.*, 2019; Liu *et al.*, 2019; Mao *et al.*, 2019; Tang *et al.*, 2021; Tang, 2022; Liu *et al.*, 2022b). In practice, PPG algorithms use a sample-based policy gradient algorithm for updating the policy parameters in the inner-loop. The higher-order meta-gradients of the sample-based inner-loop are different from those of an inner-loop that uses the expected policy gradient. Foerster *et al.* (2018a) and Rothfuss *et al.* (2019) argue that the higher-order meta-gradients of the sample-based inner-loop should approximate those of the expected inner-loop and derive alternative sample-based inner-loop update functions for that purpose. While these meta-gradient estimators are still biased, their variance has a more benign dependence on the inner-loop sample size than the unbiased meta-gradient estimator and therefore can achieve a better point in the bias-variance trade-off than either the naive or the unbiased meta-gradient estimators (Tang *et al.*, 2021; Liu *et al.*, 2022b). To further reduce the variance of this class of meta-gradient estimators, Rothfuss *et al.* (2019), Farquhar *et al.* (2019), Liu *et al.* (2019), and Mao *et al.* (2019) propose to ignore certain high-variance terms in the estimator and introduce baselines.

Alternatively, some PPG methods do not require higher-order meta-gradients. For example, Finn *et al.* (2017a) use a first-order approximation of the meta-gradient, whereas Song *et al.* (2020a) use gradient-free optimization (see Nichol *et al.*, 2018a for another first-order approximation proposed for supervised meta-learning, and recently used in meta-RL, Ren *et al.*, 2022a.) Furthermore, when meta-learning an initialization as in MAML, for a limited number of tasks or limited amount of data in the inner-loop, it may even be preferable to set the inner-loop to take no steps at all during meta-training, i.e., without adapting to each task (Gao and Sener, 2020). In this case, task adaptation occurs only through fine-tuning at test-time. The lack of explicit meta-learning can be seen as a limitation of the meta-learning approach and is discussed further in Section 6. Additionally, it is possible to use a value-based algorithm in the outer-loop, instead of a policy gradient algorithm, which avoids meta-gradients altogether (Sung *et al.*, 2017).

Outer-loop algorithms While most PPG methods use a policy-gradient algorithm in the outer-loop, other alternatives are possible (Sung *et al.*, 2017; Mendonca *et al.*, 2019). For example, one can train a critic, $Q_\theta(s, a, \mathcal{D})$, using-TD error in the outer-loop, then reuse this critic in the inner-loop (Sung *et al.*, 2017). Alternatively, instead of estimating meta-gradients via backpropagation, *evolution strategies* (ES) (Rechenberg, 1971; Wierstra *et al.*, 2014; Salimans *et al.*, 2017) may be used (Song *et al.*, 2020a). Additionally, one can train task-specific experts and then use these for imitation learning in the outer-loop. While neither directly learn exploratory behavior by optimizing Equation 2.3, they can work in practice. We provide a more complete discussion of outer-loop supervision in Section 3.6.

PPG trade-offs One of the primary benefits of PPG algorithms is that they produce an inner-loop that converges to a locally optimal policy, even when relatively few samples are available for meta-training or when the task distribution differs from meta-training to meta-testing. PPG inner-loops are generally guaranteed to converge under the same assumptions as standard policy gradient methods. For example, the MAML inner-loop converges with the same guarantees as REINFORCE (Williams, 1992), since it simply runs REINFORCE from a meta-learned initialization. Even convergence bounds are possible (Fallah *et al.*, 2021). However, as with any stochastic gradient algorithm, the guarantees given by REINFORCE are rather weak. Stochastic gradient algorithms, with a sufficiently small learning rate, only converge to a local optimum, and this can be problematic in practice in meta-RL (Xiong *et al.*, 2021). Moreover, in some cases, a step of REINFORCE can even make the policy worse on the task (Deleu and Bengio, 2018). We reproduce empirical results from Xiong *et al.* (2021) in Figure 3.2. While the PPG method, MAML, eventually converges to performant policies on many out-of-distribution tasks, it fails to do so in practice on some environments, such as those with sparse rewards. Moreover, Xiong *et al.* (2021) show that black-box methods can be made to converge similarly in practice by continuing to meta-train with gradient steps at meta-test time. Having a learning algorithm that eventually adapts to a novel task

is desirable, since it reduces the dependence on seeing many relevant tasks during meta-training.

Although parameterizing f_θ as a policy gradient method may ensure that adaptation generalizes, this structure also presents a trade-off. Typically, the inner-loop policy gradient has high variance and requires a value estimate covering the full episode, so estimating the gradient generally requires many episodes. Hence, PPG methods are generally not well-suited to few-shot problems that require stable adaption at every timestep or within very few episodes in the inner-loop. This can result in decreased performance relative to black-box methods, as seen, for example, in the sparse reward environments that require highly specialized and systematic exploration, as depicted in Figure 3.2. Moreover, PPG methods are often sample-inefficient during meta-training as well, because the outer-loop generally relies on on-policy evaluation, rather than an off-policy method that can reuse data efficiently.

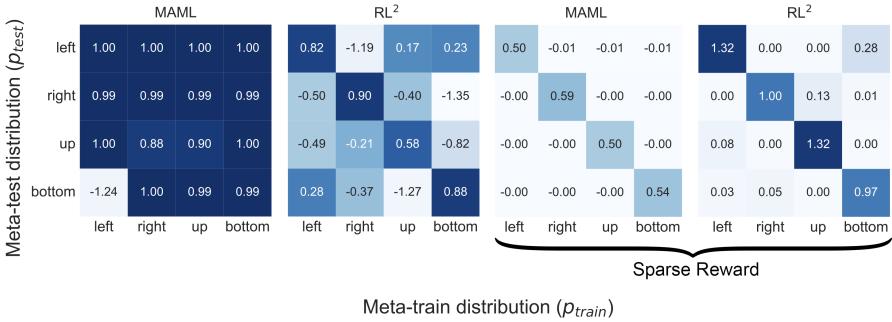


Figure 3.2: Out-of-distribution generalization metrics reproduced from Xiong *et al.* (2021). MAML and RL^2 are evaluated on variants of the environment depicted in Figure 1.1, with greater scores (darker blue) indicating a greater degree of recovered performance. While the canonical PPG method, MAML, is able to generalize better to out-of-distribution than the black-box method, RL^2 , on environments with dense reward, MAML generalizes worse on some environments in practice. MAML performs especially poorly on sparse reward environments requiring systematic exploration that is highly specific to the task distribution.

In general, there is a trade-off between generalization to novel tasks and specialization over a given task distribution. How much structure is imposed by the parameterization of f_θ determines where each algorithm lies on this spectrum. The structure of PPG methods places them near

the end of the spectrum that ensures generalization. This is visualized in Figure 3.3. While this spectrum summarizes current methods, the trade-off is not necessarily inherent to the problem setting, and future work could investigate methods that achieve the best of both worlds. In the next section, we discuss methods at the other end of the spectrum.

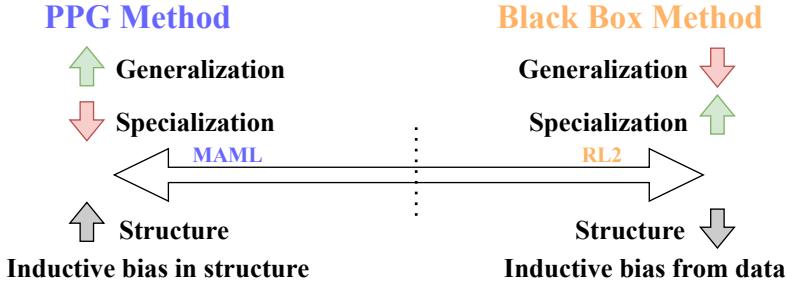


Figure 3.3: The meta-parameterization spectrum. On the left are methods like MAML that hard-code the structure of policy gradients into the inner-loop. Such methods generalize better. On the right are black box methods, which learn all of the structure of the inner-loop from data. Such methods can specialize to the task distribution better.

3.2 Black Box Methods

In contrast to PPG methods, *black box methods* are near the other end of the spectrum. Black box methods represent the inner-loop, f_θ , as a sequence model to process data for learning. Learning in the inner-loop occurs solely in the activations of this sequence model, a phenomenon sometimes called *in-context learning*. Meta-learning f_θ is a way to explicitly elicit the phenomenon of in-context learning in a sequence model. In principle, black-box methods can learn any arbitrary learning procedure, since they represent f_θ with a neural network as a universal function approximator. This places fewer constraints on the function f_θ than with a PPG method. Since f_θ represents an arbitrary function of history, the modified policy can represent an arbitrary history-dependent policy. As discussed in Section 2, such a policy is sufficient for learning an optimal policy for a POMDP, and thus for the meta-RL problem as well. Recall the RL2 algorithm (Duan *et al.*, 2016b) from Section 2.4.

In this case, f_θ is represented by a recurrent neural network, whose outputs are an input vector to the base policy. While the use of a recurrent network to output a context vector is common, black box methods may also structure the base policy and inner-loop in other ways. In this section we survey black box methods. We begin by discussing architectures that are designed for different amounts of diversity in the task distribution. Each of these architectures has different adapted policy parameters. Then, we discuss distinct ways to structure the inner-loop and alternative outer-loop algorithms. Finally, we conclude with a discussion of the trade-offs associated with black box methods.

Adapted policy parameters On one hand, some task distributions may require little adaptation for each task, as discussed in Section 3.1. For example, if a navigation task varies only by goal location, then it may be that not many policy parameters need to be adapted. Many black box methods have been applied to such task distributions (Wang *et al.*, 2016; Humplik *et al.*, 2019; Zintgraf *et al.*, 2020). In such a setting, it is sufficient to only adapt a vector, used as an input to the base policy, instead of all parameters of the base policy (Duan *et al.*, 2016b; Wang *et al.*, 2016; Zintgraf *et al.*, 2019; Humplik *et al.*, 2019; Zintgraf *et al.*, 2020; Fakoor *et al.*, 2020; Liu *et al.*, 2021). This vector, ϕ , is called a *context vector*, just as in PPG methods. This distinction is visualized in Figure 3.1. The context is produced by a recurrent neural network, or any other network that conditions on history, such as a transformer or memory-augmented network. Using a recurrent neural network, the inner-loop can be written $f_\theta(\mathcal{D}) = \text{RNN}_\theta(\mathcal{D})$. As discussed in Section 2.3, when f_θ is represented as a sequence model, f_θ and $\pi_{f_\theta(\mathcal{D})}(a|s)$ can be viewed together as a single history-dependent policy, $\pi_\theta(a|s, \mathcal{D})$. Using ϕ to represent the RNN hidden state, we can write the recurrent policy as $\pi_\theta(a|s, \phi)$. This is a common architecture for few-shot adaptation methods (Duan *et al.*, 2016b; Wang *et al.*, 2016; Humplik *et al.*, 2019; Zintgraf *et al.*, 2020; Fakoor *et al.*, 2020; Liu *et al.*, 2021).

On the other hand, some task distributions may require significant differences in behavior between the optimal policies. By conditioning a policy on a context vector, all of the weights and biases of π must generalize between all tasks. However, when significantly distinct policies

are required for different tasks, forcing base policy parameters to be shared may impede adaptation (Beck *et al.*, 2022). Alternatively, just as in PPG methods, there may be no context vector, but the inner-loop may adapt the weights and biases of the base policy, directly (Beck *et al.*, 2022). The weights and biases here are the task parameters, ϕ , output by the inner-loop. The inner-loop may produce all of the parameters of a feed-forward base policy (Beck *et al.*, 2022), or may modulate the weights of a recurrent base learner (Flennerhag *et al.*, 2020). In these cases, using one network to map all data in the trial, \mathcal{D}_j , to weights and biases of another network defines a hypernetwork (Ha *et al.*, 2017). The architecture can be written as $\pi_\phi(a|s)$, where $\phi = h_\theta(\text{RNN}_\theta(\mathcal{D}))$, h is a hypernetwork, and ϕ are the weights and biases of the policy. See Figure 3.4 for an illustration.

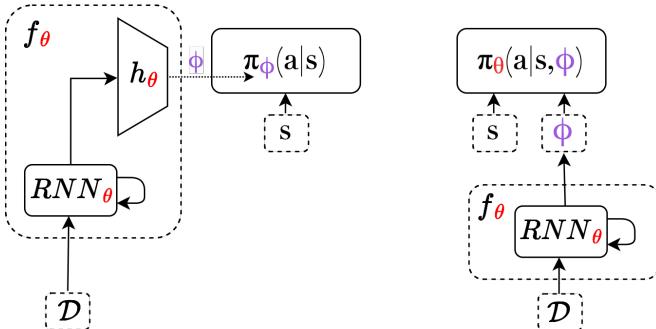


Figure 3.4: Illustration of black box meta-RL with hypernetwork (left) and with a context vector (right). When the inner-loop is a recurrent neural network, generating all the parameters of the policy, including its weights and biases, results in a hypernetwork. Alternatively, the RNN can output a context vector on which the policy is conditioned.

Inner-loop representation While many black box methods simply use recurrent neural networks as the inner-loop representation (Heess *et al.*, 2015; Duan *et al.*, 2016b; Wang *et al.*, 2016; Fakoor *et al.*, 2020), alternative representations have also been investigated. Some of these representations are biologically inspired (Bellegc *et al.*, 2018; Miconi *et al.*, 2018). One of the most common biologically inspired representations

leverages Hebbian learning (Miconi *et al.*, 2018; Miconi *et al.*, 2019; Najarro and Risi, 2020; Chalvidal *et al.*, 2022; Rohani *et al.*, 2022). Hebbian learning (Hebb, 1949) is a biologically inspired method in which weight updates are a function of the associated activations in the previous and next layers. The update to the weight ($w_{i,j}^k$) from the i th activation in layer k (x_i^k), to the j th activation in layer $k+1$ (x_j^{k+1}) generally has the form

$$w_{i,j}^k := w_{i,j}^k + \alpha(ax_i^k x_j^{k+1} + bx_i^k + cx_j^{k+1} + d),$$

where α is a learning rate and α, a, b, c, d are all meta-learned parameters of the inner-loop learned in θ . While Hebbian learning is one of the most common biologically inspired representations, most methods aligned with recent machine learning literature use some form of attention (Graves *et al.*, 2014; Vaswani *et al.*, 2017) to parameterize the inner-loop (Oh *et al.*, 2016; Ritter *et al.*, 2018b; Mishra *et al.*, 2018; Fortunato *et al.*, 2019; Ritter *et al.*, 2021; Wang *et al.*, 2021; Emukpere *et al.*, 2021; Melo, 2022; Xu *et al.*, 2022; Team *et al.*, 2023; Elawady *et al.*, 2024).

Attention can be thought of as a soft form of a key-value lookup in a dictionary. Specifically, it is a mechanism for combining different vectors based on the similarity of their associated key vectors to a given query vector. Given a query vector, q , a matrix, V , where each row is one of n value vectors over which to attend, and a matrix K , where each row is one of n key vectors, then attention can be written

$$\text{Attention}(q, K, V) = \sum_{i=0}^{i=n} w_i(q, K_i) V_i$$

where $w \in \Delta^n$ is a weight vector defining the convex combination of value vectors (Graves *et al.*, 2014). Generally, attention computes the weights as $w = \text{softmax}(Kq)$, leading to an attention mechanism that simplifies to

$$\text{Attention}(q, K, V) = V^T \text{softmax}(Kq) = (\text{softmax}(q^T K^T)V)^T.$$

Additionally, computing multiple queries, with each as a row vector in Q , we can write this as

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V.$$

Some methods combine attention with convolution (Mishra *et al.*, 2018), or use attention over past recurrent states (Ritter *et al.*, 2018b; Fortunato *et al.*, 2019; Team *et al.*, 2023; Oh *et al.*, 2016), while others use self-attention alone (Ritter *et al.*, 2021; Wang *et al.*, 2021; Team *et al.*, 2023). For example, to attend to past recurrent states, q may be a function of the current hidden state of a recurrent network, while K and V may be (two different linear projections of) all prior hidden states computed over \mathcal{D} . A generalization of such methods can be seen in Figure 3.5. In contrast, self-attention models Q , K , and V all as linear projections of the inputs in \mathcal{D} first, then as linear projections of the previous attention layer, for multiple attention layers in a row.

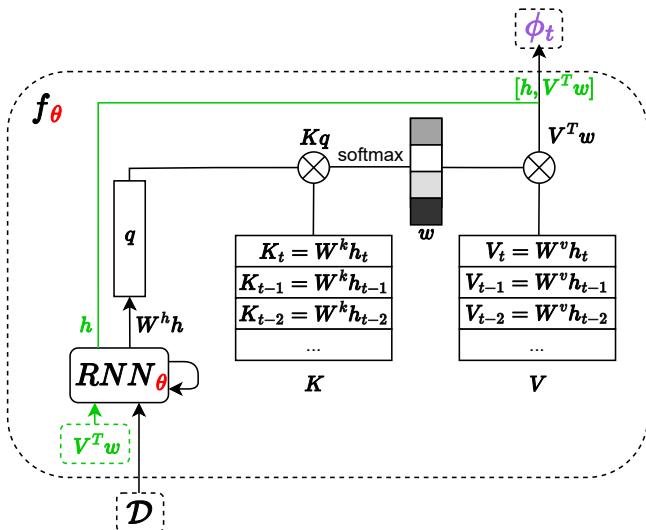


Figure 3.5: Attention over past recurrent states in the inner-loop of black box methods. One approach is to use the current hidden state as a query in attention over past recurrent states. The keys (K) and values (V) are all linear projections of the past recurrent states. The output, ϕ , is a convex combination of the projected hidden states (V), where the combination is specified by this weight vector (w), computed from the similarity between the keys (K) and query (q). Components in green are optional. In order to provide long-term context to the RNN, the output from attention over past hidden states, $V^T w$, can be passed as an input to the RNN at the next timestep. Additionally, the output, ϕ , can be $V^T w$, h , or a concatenation of both. Passing the RNN state, h , as an output, with $V^T w$ as an input, allows multiple steps of attention to be integrated into the final output.

Attention mechanisms seem to aid in generalization to novel tasks outside of the distribution, $p(\mathcal{M})$, (Fortunato *et al.*, 2019; Melo, 2022) and self-attention may be useful for complex planning (Ritter *et al.*, 2021). Still, attention is computationally expensive: whereas recurrent networks use $O(1)$ memory and compute per timestep, attention generally requires $O(t^2)$ memory and compute per timestep t , which may cross many episodes. While fast approximations of attention exist (Katharopoulos *et al.*, 2020), solutions in meta-RL often simply maintain only memory of a fixed number of the most recent transitions (Mishra *et al.*, 2018; Fortunato *et al.*, 2019). Nonetheless, both transformers, which use self-attention, and recurrent neural networks can still struggle to meta-learn simple inductive biases, particularly for complex task distributions generated by simple underlying rules in low-dimensional spaces (Kumar *et al.*, 2020; Chan *et al.*, 2022).

Outer-loop algorithms While many black box methods use on-policy algorithms in the outer-loop (Duan *et al.*, 2016b; Wang *et al.*, 2016; Zintgraf *et al.*, 2020), it is straightforward to use off-policy algorithms (Rakelly *et al.*, 2019; Fakoor *et al.*, 2020; Liu *et al.*, 2021), which bring increased sample efficiency to RL. For discrete actions, it is also straightforward to use Q -learning (Fakoor *et al.*, 2020; Liu *et al.*, 2021; Zhang and Kan, 2022), in which case the inner-loop must change as well. In this case, the inner-loop estimates Q -values instead of directly parameterizing a policy. The policy can then act greedily with respect to these Q -values at meta-test time. This approach can be thought of as modifying recurrent Q -networks (Hausknecht and Stone, 2015) to fit the meta-RL setting, which compares favorably compared to other state-of-the-art meta-RL methods (Fakoor *et al.*, 2020).

Black box trade-offs One key benefit of black box methods is that they can rapidly alter their policies in response to new information, whereas PPG methods generally require multiple episodes of experience to get a sufficiently precise inner-loop gradient estimate. For example, consider an agent that must learn which objects in a kitchen are hot at meta-test time. While estimating a policy gradient, a PPG method may touch a hot stove multiple times before learning not to. In contrast, a

black box method may produce an adaptation procedure that never touches a hot surface more than once. Black box methods can learn such responsive adaptation procedures because they represent the inner-loop as an arbitrary function that maps from the cumulative task experience to the next action.

However, black-box methods also present a trade-off. While black-box methods can tightly fit their assumptions about adaptation to a narrow distribution of data, $p(\mathcal{M})$, increasing specialization, they often struggle to generalize outside of $p(\mathcal{M})$ (Wang *et al.*, 2016; Finn and Levine, 2018; Fortunato *et al.*, 2019; Xiong *et al.*, 2021). Consider the robot chef: while it may learn to not touch hot surfaces, it is unlikely a black box robot chef will learn a completely new skill, such as how to use a stove, if it has never seen a stove during meta-training. In contrast, a PPG method could still learn such a skill at meta-test time with sufficient data. While there have been efforts to both broaden the set of meta-training tasks and manually add inductive biases to black-box methods for generalization, which we discuss in Section 4, whether to use a black-box method or PPG method generally depends on the amount of generalization and specialization in the problem to be solved.

Additionally, there exist trade-offs in outer-loop optimization challenges between PPG and black box methods. On one hand, as discussed in Section 3.1, PPG methods often estimate a meta-gradient, which is difficult to compute (Al-Shedivat *et al.*, 2018a), especially for long horizons (Wu *et al.*, 2018). On the other hand, black box methods do not have the structure of optimization methods build into them, so they can be harder to train from scratch, and have associated outer-loop optimization challenges even for short horizons. Black box methods generally make use of recurrent neural networks, which can suffer from vanishing and exploding gradients (Pascanu *et al.*, 2013). Moreover, the optimization of recurrent neural networks can be especially difficult in reinforcement learning (Beck *et al.*, 2020), while transformers can be even more problematic to train (Parisotto *et al.*, 2020; Melo, 2022). While large transformers have shown some notable success in meta-RL, such solutions require the use of curriculum learning and distillation to train stably (Team *et al.*, 2023). Thus, the rapid updates of black

box methods in meta-RL, while enabling fast learning, also present challenges for meta-learning.

Some methods use both PPG and black-box components (Vuorio *et al.*, 2019; Xiong *et al.*, 2021; Ren *et al.*, 2022a). In particular, even when training a fully black-box method, the policy or inner-loop can be fine-tuned with policy gradients at meta-test time (Lan *et al.*, 2019; Xiong *et al.*, 2021; Imagawa *et al.*, 2022).

3.3 Task Inference Methods

Closely related to black box methods are *task inference* methods, which often share the same parameterization as black box methods and thus can be considered a subset of them. However, parameterizations of the inner-loop may be specific to task inference methods (Rakelly *et al.*, 2019; Korshunova *et al.*, 2020; Zintgraf *et al.*, 2020), which generally train the inner-loop to perform a different function by optimizing a different objective.

Task inference methods generally aim to identify the MDP, or task, to which the agent must adapt, in the inner-loop. In meta-RL, the agent must repeatedly adapt to an *unknown* MDP whose representation is not given as input to the inner-loop. While the agent ultimately acts to maximize reward, the entire purpose of the inner-loop can be described as identifying the task. The agent’s belief about what it should do can be represented as a distribution over tasks. As discussed in Section 2, this posterior distribution constitutes a sufficient statistic, or information state (Subramanian *et al.*, 2022), for the meta-RL problem. Since we already know the form of this sufficient statistic, the inner-loop can model it directly, instead of learning a mapping from history to action end-to-end. *Task inference* is the process of inferring this posterior distribution over tasks, conditional on what the agent has seen so far.

Consider the case where the agent has uniquely identified the task. Then, at this point, the agent knows the MDP and could in fact use classical planning techniques, such as value iteration, to compute the optimal policy directly. In this scenario, no further learning or data collection is required. More practically, if the task distribution is reasonably small and finite, we can avoid even having to explicitly plan,

by learning a mapping from the task to the optimal policy directly, during meta-training. In fact, training a policy over a distribution of tasks, with the policy conditioned on the true task, can be taken as the definition of multi-task RL (Yu *et al.*, 2020a). In the multi-task case, a mapping is learned from a known task to a policy. In meta-RL the only difference is that the task is not known. Thus task inference can be seen as an attempt to move a meta-learning problem into the easier multi-task setting.

When uncertainty remains in the distribution, instead of mapping a task to a base policy, task inference methods generally map a task distribution, given the current data, to a base policy. This can be seen as learning a policy conditioned on a (partially) inferred task. In this case, learning becomes the process of reducing uncertainty about the task. The agent must collect data that enables it to identify the task. That is, the agent must explore to reduce uncertainty in the posterior given by task inference. Task inference is therefore a useful way to frame exploration, and many task inference methods are framed as tools for exploration, which we discuss in Section 3.4. Optimally handling uncertainty in the task distribution is difficult, and is discussed in Section 3.5.

In this Section we discuss two methods for task inference that use supervised learning but also require assumptions about the information available for meta-training. We then discuss alternative methods without such assumptions, and how the inner-loops are usually represented. Finally, we conclude with a discussion of the trade-offs concerning task inference methods. Table 3.1 summarizes these categories and task inference methods.

Task inference with privileged information A straightforward method for inferring the task is to add a supervised loss so that a black box f_θ predicts some estimate of the task, $\hat{c}_\mathcal{M}$, given some known representation of the task, $c_\mathcal{M}$ (Humplik *et al.*, 2019). For example, a recurrent network may predict the task representation conditional on all data collected so far. Recall that ϕ , the task parameters, are the adapted policy parameters output by the inner-loop. Most commonly, ϕ is passed directly to the policy as an input vector: $\pi_\theta(a|s, \phi)$. In task inference,

the vector ϕ is generally the predicted task estimate: $\pi_\theta(a|s, \hat{c}_\mathcal{M})$, where $\phi = \hat{c}_\mathcal{M}$. For computing the supervised loss, this task representation must be known during meta-training, and so constitutes a form of *privileged information*. The representation may, for instance, be a one-hot representation of a task index, if the task distribution is discrete and finite. Or, it may be some parameters defining the MDP. For example, if kitchens differ in the location of the stove and refrigerator, the task representation, $c_\mathcal{M}$, could be a vector of all these coordinates. In this case, f_θ would predict these coordinates. The representation may even contain sub-tasks and their hierarchies, or human preferences (Ren *et al.*, 2022b), when such information is known (Sohn *et al.*, 2020; Zhang and Kan, 2022), or make use of known transition functions to explicitly approximate a Bayesian posterior over the tasks (Lee *et al.*, 2019). Commonly, the task-inference objective, denoted here as J_{infer} , is given by the maximum likelihood estimate:

$$J_{\text{infer}}(\theta) = \mathbb{E}_{\mathcal{M}}[\mathbb{E}_{\mathcal{D}|\pi}[\log p_\theta(c_\mathcal{M})]].$$

When passing the task to the policy, there are a few important representation choices. First, instead of conditioning the policy on the task representation directly, we can pass a representation with more information about task uncertainty. This can be accomplished, for instance, by passing to the policy the penultimate layer when predicting $\hat{c}_\mathcal{M}$ (Humplik *et al.*, 2019). Then the task parameters, ϕ , passed to the policy, are trained by inferring $\hat{c}_\mathcal{M}$, but only after a subsequent linear transformation: $\hat{c}_\mathcal{M} = L_\theta(\phi)$. This hidden layer may contain more information than $\hat{c}_\mathcal{M}$, and $\hat{c}_\mathcal{M}$ can always be computed from it by the policy. Second, it can be useful to add a stop-gradient to ϕ before passing it to the policy (Humplik *et al.*, 2019; Zintgraf *et al.*, 2021b), to prevent conflicting gradients. Finally, when training f_θ with J_{infer} , as a supervised objective, the dataset \mathcal{D} may contain off-policy data without bias, which may give a particular advantage in the off-policy setting, as compared to J_{meta} given by Equation 2.3 (Humplik *et al.*, 2019).

Task inference with multi-task training Some research uses the multi-task setting to improve task inference with privileged information (Humplik *et al.*, 2019; Kamienny *et al.*, 2020; Raileanu *et al.*, 2020;

Liu *et al.*, 2021; Peng *et al.*, 2021). The task representation may contain little task-specific information (e.g., if it is one-hot representation) or task-specific information that is irrelevant to the policy (e.g. the amount of oxygen in the kitchen). For example, consider the more concrete task of navigation to a goal on the perimeter of a circle, as discussed in Section 1. In this case, if the task representation is one-hot, it may be useful to instead have access to the $(x_{\text{goal}}, y_{\text{goal}})$ coordinates of the goal. Additionally, if the task representation contains the location of an additional irrelevant object, $(x_{\text{goal}}, y_{\text{goal}}, x_{\text{object}}, y_{\text{object}})$, then it may be useful to instead have access to a more parsimonious task descriptor, $(x_{\text{goal}}, y_{\text{goal}})$, that contains the goal location alone. In general, the entire MDP does not need to be uniquely identified. The agent only needs to identify the variations between MDPs that occur in the task distribution, $p(\mathcal{M})$. Even more specifically, the agent only needs to identify the subset of those variations that change the optimal policy.

To address uninformative and irrelevant task information, representations can be learned by pre-training in the multi-task setting (Humphlik *et al.*, 2019; Liu *et al.*, 2021). Let $g_\theta(c_M)$ be a function that encodes the task representation. First an informed policy, $\pi_\theta^{\text{multi}}(a|s, g_\theta(c_M))$, can be trained. This is the multi-task phase, and enables the learning of g_θ . Since this representation, $g_\theta(c_M)$, is learned end-to-end, it contains the information relevant for solving the task. For example, g_θ could transform c_M from $(x_{\text{goal}}, y_{\text{goal}}, x_{\text{object}}, y_{\text{object}})$ to $(x_{\text{goal}}, y_{\text{goal}})$. Often, an information bottleneck (Alemi *et al.*, 2017) is used to ensure it contains only this information (Humphlik *et al.*, 2019; Liu *et al.*, 2021). After this, the inner-loop can infer the learned representation, $g_\theta(c_M)$, from the meta-trajectory, which we write as $\hat{g}_\theta(\mathcal{D})$ (see Figure 3.6). For example, in the circle navigation task, after learning the coordinate representation, $(x_{\text{goal}}, y_{\text{goal}})$, the inferred \hat{g}_θ could be represented by the inferred coordinates, $(\hat{x}_{\text{goal}}, \hat{y}_{\text{goal}})$. Alternatively, the informed multi-task RL may be performed concurrently with the meta-RL (Kamienny *et al.*, 2020).

For example, the informed policy regularization algorithm (IMPORT, Kamienny *et al.*, 2020) follows the simultaneous-training paradigm. In this case, the auxiliary inference objective is given by

$$J_{\text{infer}}(\theta) = \mathbb{E}_{\mathcal{M}}[\mathbb{E}_{\mathcal{D}|\pi}[(g_\theta(c_M) - \hat{g}_\theta(\mathcal{D}))^2]].$$

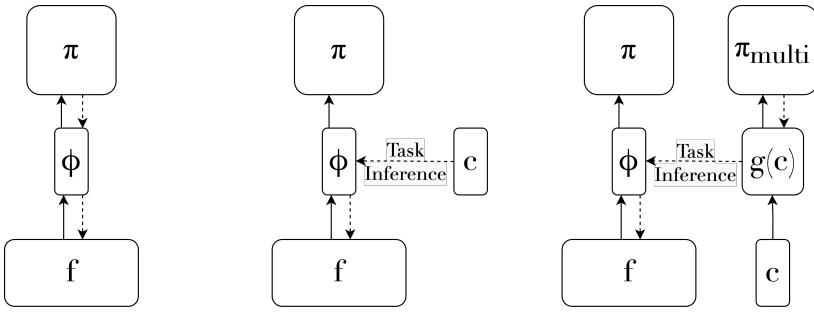


Figure 3.6: Illustration of normal meta-RL (left), task inference using privileged information (middle), task inference using multi-task training (right). Solid arrows represent forward propagation, and dashed arrows represent backpropagation. Task inference with privileged information uses a true task representation, c , and then backpropagates that inference to train ϕ . Task inference with multi-task training learns an encoding of the task, $g(c)$, to maximize the return of an informed policy, π_{multi} , then uses inference of this learned encoding to train ϕ .

This inference objective forces the encoding of g and \hat{g} to look similar. While optimizing this objective, the algorithm also optimizes the normal meta-learning objective (see Equation 2.3) but with the policy conditioned on the inferred task representation: $\pi_\theta(a|s, \hat{g}_\theta(\mathcal{D}))$. Additionally, IMPORT simultaneously optimizes the meta-RL objective but with the multi-task policy conditioned on the learned task representation: $\pi_\theta^{\text{multi}}(a|s, g_\theta(c_M))$.

Some task distributions even allow for significant shared behavior between the informed multi-task agent and the uninformed meta-RL agent. This sharing is generally possible when little exploration is needed for the meta-RL policy to identify the task. In this case instead of only inferring the privileged task information, the meta-RL agent may imitate the multi-task agent through distillation (Weihs *et al.*, 2021; Nguyen *et al.*, 2022), which we cover in Section 3.6, or by direct parameter sharing of policy layers (Kamienny *et al.*, 2020; Peng *et al.*, 2021). For instance, in the IMPORT algorithm, π_θ and $\pi_\theta^{\text{multi}}$ share parameters by using the same components of the meta-parameters vector, θ . In this case, the entire policy is shared, such that $\pi = \pi^{\text{multi}}$. Even when the multi-task and meta-RL policies are computed sequentially, the pre-trained multi-task agent may still be used as an initialization for the meta-RL agent (Beck *et al.*, 2023).

In contrast, when task distributions require taking sufficiently many exploratory actions to identify the task, sharing policies becomes less feasible. For example, in the circle navigation task, the informed multi-task policy, with access to the goal, never needs to explore, whereas the primary behavior of the meta-RL policy is exploration around the circumference of the circle. If one were to reuse the multi-task policy conditioned on the inferred goal, $\pi_\theta = \pi_\theta^{\text{multi}}(a|s, \hat{x}_{\text{goal}}, \hat{y}_{\text{goal}})$, as the meta-RL policy, then the inference would not have sufficient data to be accurate. The initial inferred goal location, for example, may be $(0, 0)$, which is not in the training distribution of the multi-task agent. In this case, the behavior of the multi-task agent would be undefined. However, sharing some parameters, simultaneously training the shared policy on inferred representation (Kamienny *et al.*, 2020), or fine-tuning the shared policy as an initialization for the meta-RL policy (Beck *et al.*, 2023), can solve this issue. For example, if training the shared policy simultaneously with known and inferred goal locations, $(x_{\text{goal}}, y_{\text{goal}})$ and $(\hat{x}_{\text{goal}}, \hat{y}_{\text{goal}})$, the agent can learn to execute exploratory behavior whenever the inferred task is $(0, 0)$, or has high uncertainty in the posterior distribution. Still, collecting data sufficient for inferring the task correctly may be difficult. Often, intrinsic rewards are additionally needed to even be able to collect the data enabling task inference. Such exploration in task inference methods is discussed in Section 3.4.

Task inference without privileged information Other task inference methods do not rely on privileged information in the form of the known task representation. If the algorithm is allowed to know whether two trajectories were generated by the same task, then one option is to use this label as a learning signal for the encoder (Fu *et al.*, 2021; Mu *et al.*, 2022; Choshen and Tamar, 2023; Guo *et al.*, 2018; Luo *et al.*, 2022). More commonly, the representation can be (a sample from) a latent variable parameterizing a learned reward function or transition function (Zhou *et al.*, 2019; Zintgraf *et al.*, 2020; Zhang *et al.*, 2021; Zintgraf *et al.*, 2021b; He *et al.*, 2022). These methods use only information already observable and train $f_\theta(\mathcal{D})$ to represent a task distribution, given trajectories in \mathcal{D} .

For example, Zintgraf *et al.* (2020) propose to learn the task parameters, ϕ , as a latent variable parameterizing the reward and transition function. In this case, the latent is trained in a self-supervised manner by reconstructing trajectories. This is accomplished using only observable information by predicting rewards and next states for each transition, given each ϕ_t :

$$\begin{aligned} J_{\text{infer}}(\theta) &= \mathbb{E}_{\mathcal{M}}[\mathbb{E}_{\mathcal{D}|\pi}[\sum_{t=0}^{HN-1} \log p_{\theta}(\mathcal{D}|\phi_t)]] \\ &= \mathbb{E}_{\mathcal{M}}[\mathbb{E}_{\mathcal{D}|\pi}[\sum_{t=0}^{HN-1} \sum_{s,a,r,s' \in \mathcal{D}} [\log p_{\theta}(s'|s,a,\phi_t) + \log p_{\theta}(r|s,a,\phi_t)]]], \end{aligned}$$

where the inner summation is taken over consecutive transitions, (s, a, r, s') , in \mathcal{D} . An extension of this algorithm has even been proposed with a hierarchical latent variable to accommodate the additional structure in distributions of POMDPs (Akuzawa *et al.*, 2021).

Inner-loop representation Generally, task inference is implemented by adding an additional loss function, and not by any particular meta-parameterization of f_{θ} . While task inference methods do not require a particular meta-parameterization, most implementations use a “black box,” such as a recurrent neural network (Humplik *et al.*, 2019; Zintgraf *et al.*, 2020; Kamienny *et al.*, 2020; Liu *et al.*, 2021; Zintgraf *et al.*, 2021c). Since many task inference methods infer a latent variable, it is also common for f_{θ} to explicitly model this distribution using a variational information bottleneck (Humplik *et al.*, 2019; Rakelly *et al.*, 2019; Zintgraf *et al.*, 2020; Liu *et al.*, 2021; Zintgraf *et al.*, 2021c).

In this case, the variational distribution defining the latent variable conditions on \mathcal{D} . We write this as $q_{\theta}(z|\mathcal{D})$. This variational distribution has its own parameters inferred from the dataset, such as a mean, $\mu_{\theta}(\mathcal{D})$, and a covariance matrix $\text{diag}(\sigma_{\theta}(\mathcal{D}))$. First, the task representation is reconstructed, e.g., with a linear projection of samples from q_{θ} : $\hat{c}_{\mathcal{M}} = L_{\theta}^c(z \sim q_{\theta})$. Then, the inference is trained by maximizing J_{infer} , also called the reconstruction loss, over samples from the variational distribution:

$$J_{\text{infer}}(\theta) = \mathbb{E}_{\mathcal{M}}[\mathbb{E}_{\mathcal{D}|\pi}[p_{\theta}(c_{\mathcal{M}}|\hat{c}_{\mathcal{M}})]],$$

along with a KL-Divergence to a prior,

$$J_{\text{prior}}(\theta) = \mathbb{E}_{\mathcal{M}}[\mathbb{E}_{\mathcal{D}|\pi}[D(q_\theta(z)||p_\theta(z))]].$$

Once the task inference is trained, the parameters passed to the policy, ϕ , must be chosen. One option is to represent ϕ as a projection of the full distribution of the latent variable, e.g., its mean and variance, in order to capture uncertainty about the task (Zintgraf *et al.*, 2020; Imagawa *et al.*, 2022). This can be written $\pi_\theta(a|s, L_\theta^\phi(\mu, \sigma))$, where $\phi = L_\theta^\phi(\mu, \sigma)$. Combining this representation of the latent distribution with the J_{infer} for trajectory reconstruction above, we arrive at a canonical method, variational Bayes-Adaptive Deep RL (VariBAD, Zintgraf *et al.*, 2021b). This method is depicted in Figure 3.7.

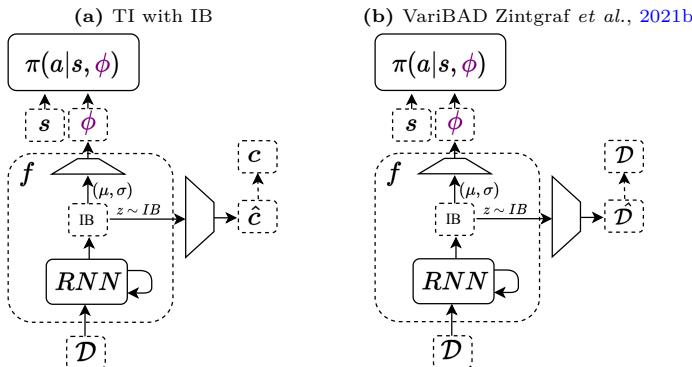


Figure 3.7: Task inference with an information bottleneck (IB) (a), task inference with an IB and without privileged information, as in Zintgraf *et al.* (2021b) (b). In either case, the parameters of a latent distribution are passed to the policy. When using privileged information, the task representation, c , is reconstructed from samples of this distribution; otherwise, the set of trajectories, \mathcal{D} , is reconstructed.

Alternatively, it is possible for the policy to condition on samples from the information bottleneck (i.e., $\pi_\theta(a|s, z \sim q_\theta)$, where $\phi = z$), instead of conditioning on the parameters of the information bottleneck directly (Rakelly *et al.*, 2019). In this case, the task inference can be trained entirely from the actor and critic loss given by the meta-learning objective, rather than a distinct task inference objective. Rakelly *et al.* (2019) and Wen *et al.* (2022) use such probabilistic embeddings for

actor-critic RL, as introduced by the PEARL method (Rakelly *et al.*, 2019).

Several methods also make use of the exchangeability, or permutation invariance, of the transitions implied by the Markov property in task inference (Rakelly *et al.*, 2019; Korshunova *et al.*, 2020; Raileanu *et al.*, 2020; Imagawa *et al.*, 2022; Beck *et al.*, 2024). PEARL specifically models the task distribution conditioned on \mathcal{D} as a product of individual distributions conditioned on each transition in \mathcal{D} :

$$z \sim q_\theta(z|\mathcal{D}) \propto \prod_{s,a,r,s' \in \mathcal{D}} q_\theta(z|s,a,r,s').$$

While it is not necessary to model this permutation invariance, standard sequence models are sensitive to the ordering of their inputs, which can be detrimental to learning when the order does not matter (Beck *et al.*, 2020). For this reason, it may be useful to embed the structure of permutation invariance into the task inference model as an inductive bias. Specifically, PEARL uses a product of Gaussians, $q_\theta(z|s,a,r,s') = \mathcal{N}(z; \mu_\theta(s,a,r,s'), \text{diag}(\sigma_\theta(s,a,r,s')))$, which has a simple closed form for the joint mean and covariance. However, other methods forgo the product entirely, and use more general permutation invariant representations, such as neural processes (Garnelo *et al.*, 2018; Wang and Hoof, 2022) and transformers (Vaswani *et al.*, 2017).

Finally, some work uses hypernetworks in the inner-loop (Peng *et al.*, 2021; Beck *et al.*, 2022; Beck *et al.*, 2023). It is possible to use a neural network to map an inferred task to base-network weights and biases: $\pi_{h_\theta(\hat{c}_\mathcal{M})}(a|s)$ (Beck *et al.*, 2022). In this case, the mapping is a hypernetwork, and it can be trained end-to-end with the meta-learning objective. Alternatively, it is also possible to use the weights and biases of existing experts trained in the multi-task setting as explicit targets for the hypernetwork. In this case, the using these targets for direct supervision can be seen as (additionally) training with a task-inference objective (Peng *et al.*, 2021; Beck *et al.*, 2023). Representing the task as the weights and biases of the expert policies has only been investigated as supervision for hypernetworks, but it could, in principle, apply as a task representation for any task-inference method. Note that even when the hypernetwork has no direct supervision, it is still possible to

make use of pre-training in the multi-task setting. For example, the meta-learned hypernetwork can be pre-trained in the multi-task setting (Beck *et al.*, 2023).

Task inference trade-offs Task inference methods present trade-offs in comparison to other methods. First, we consider the comparison to PPG methods then we consider the comparison to black box methods. In comparison to PPG methods, task inference methods impose less structure on the inner-loop. On the one hand, PPG methods generalize well to novel tasks because of the additional structure. In the case where a novel task cannot be represented using the task representations learned during meta-training, task inference methods fail (Rimon *et al.*, 2022), whereas PPG methods generally adapt to the novel task using a policy gradient. On the other hand, PPG methods are unlikely to recover an algorithm as efficient as task inference. For distributions where task inference is possible, fitting such a method to the task distribution may enable faster adaptation. If there are not many tasks in the distribution they are easily inferable from a few consecutive transitions, inferring a latent task using a task inference method can be more sample efficient than learning a new policy with a PPG method. This is the trade-off between generalization and specialization depicted in Figure 3.3.

In comparison to black box methods, task inference methods impose more structure. On the one hand, task inference methods often add additional supervision through the use of privileged information (Humplik *et al.*, 2019; Kamienny *et al.*, 2020; Liu *et al.*, 2021) or the use of self-supervision (Zintgraf *et al.*, 2020; Zintgraf *et al.*, 2021c), which may make meta-training more stable and efficient (Humplik *et al.*, 2019; Zintgraf *et al.*, 2020). This is particularly useful since recurrent policies, often used in task inference and black box methods, are difficult to train in RL (Beck *et al.*, 2020). On the other hand, there is evidence that black-box training can be stabilized in meta-RL with the use of a hypernetwork architecture and reasonable initialization (Beck *et al.*, 2023), and training the inner-loop to accomplish any objective that is not Equation 2.3 may be suboptimal with respect to that meta-RL objective over the given task distribution. An empirical comparison between black-box methods and task-inference methods can be seen in

Table 3.2: Average return across continuous control environments reproduced from Zintgraf *et al.* (2021b) and Beck *et al.* (2023). We compare the black-box method, RL^2 , and task-inference method, VariBAD, with and without hypernetworks. All methods are evaluated over the training distribution and evaluated after a single episode for adaptation (0-shot with $H = 1$). Results are rounded to the nearest hundred. We see that the task-inference method generally outperforms the black-box method without hypernetworks, but with hypernetworks, the black-box methods perform equivalently or better. (Note that the Walker environment is excluded from this table, since the experimental setups on that environment differ between Zintgraf *et al.* (2021b) and Beck *et al.* (2023). PPG methods do not compare favorably in this setting; for PPG comparisons, see Figure 3.2 and Table 3.3.)

	Ant-Dir	Cheetah-Dir	Cheetah-Vel
RL^2	1,200	1,200	0
VariBAD	1,300	2,000	0
RL^2 with Hypernetwork	1,400	2,400	0
VariBAD with Hypernetwork	1,400	2,200	0

Table 3.2. Beyond comparisons to PPG methods and black box methods, task inference methods provide some additional advantages. Task inference methods that model the reward and dynamics can be used to sample additional (imagined) tasks (Rimon *et al.*, 2022), which can be seen as a type of model-based meta-RL, discussed later in Section 3.7. And, as we show in Section 3.4, task inference methods also are useful for exploration.

3.4 Exploration and Meta-Exploration

Exploration is the process by which an agent collects data for learning. In standard RL, exploration should work for any MDP and may consist of random on-policy exploration, epsilon-greedy exploration, or methods to find novel states. In meta-RL, this type of exploration still occurs in the outer-loop, which is called *meta-exploration*. However, there additionally exists exploration in the inner-loop, referred to as just *exploration*, which is where we begin our discussion. This inner-loop exploration is specific to the distribution of MDPs, $p(\mathcal{M})$. To enable sample efficient adaptation, the meta-RL agent uses knowledge about the task distribution to explore efficiently. For instance, instead of taking random actions, the robot chef may open every cabinet to learn

Table 3.3: Average return across continuous control environments reproduced from Zintgraf *et al.* (2021b). We compare the black-box method, RL^2 , and E-MAML Stadie *et al.*, 2018, a variant of the PPG method, MAML, designed to encourage exploration. Results are rounded to the nearest hundred. (See Zintgraf *et al.* (2021b) for details.) RL^2 and E-MAML are evaluated over the training distribution and evaluated after a single episode for adaptation (0-shot with $H = 1$), which favors black-box methods. Note that even E-MAML after 19 episodes of adaptation (19-shot with $H = 20$) is still outperformed by black-box methods on these domains. (For environments where PPG methods are favorable, see the out-of-distribution evaluations in Figure 3.2.)

	Ant-Dir	Cheetah-Dir	Cheetah-Vel	Walker
$\text{RL}^2 (H = 1)$	1,200	1,200	0	300
E-MAML ($H = 1$)	300	0	-200	200
E-MAML ($H = 20$)	300	300	-100	200

about the location of food items and utensils when first entering a new kitchen. This exploration is targeted and used to provide informative trajectories in \mathcal{D} that enable few-shot adaptation to the MDP within the task distribution.

Recall that in the few-shot adaptation setting, on each trial, the agent is placed into a new task and is allowed to interact with it for a few episodes (i.e., its few shots K), before being evaluated on solving the task in the next few episodes (i.e., over the $H - K$ episodes in Equation 2.3). An illustration can be seen in Figure 3.8. Intuitively, the agent must explore to gather information during the first few shots that enables it to best solve the task in later episodes. More generally, there is an exploration-exploitation trade-off, where the agent must balance taking exploratory actions to learn about the new task (potentially even beyond the initial few shots) with exploiting what it already knows to achieve high rewards. It is always optimal to explore in the first K episodes, since no reward is given to the agent. However, the optimal amount of exploration in the remaining $H - K$ shots depends on the size of the evaluation period $H - K$: When $H - K$ is large, more exploration is optimal, as sacrificing short-term rewards to learn a better policy for higher later returns pays dividends, while when $H - K$ is small, the agent must exploit more to obtain any reward it can, before time runs out. In this section, we survey approaches that navigate this trade-off. Table 3.4 summarizes these categories. As discussed in Section 2, there

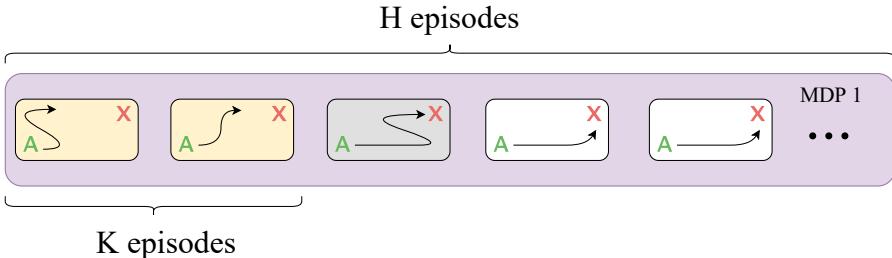


Figure 3.8: Illustration of “free” exploration in first K episodes (yellow), followed by not free exploration (gray), followed by exploitation (white). An agent (A) is trying to identify how to get to a goal location (X). The agent has K shots, or free episodes to explore. However, K episodes may not be enough, so in the $(K + 1)$ th episode, the agent is still exploring the map to find the goal, and is penalized for this. In the remaining two episodes, the agent has learned to navigate to the goal optimally, once the goal has been found, and no longer needs to explore.

is an optimal solution to this exploration problem that maximizes the meta-RL objective. In the next section, we discuss a framework that formalizes making this trade-off optimally.

End-to-end optimization Perhaps the simplest approach is to learn to explore and exploit *end-to-end* by directly maximizing the meta-RL objective (Equation 2.3) as done by black box meta-RL approaches (Duan *et al.*, 2016b; Wang *et al.*, 2016; Mishra *et al.*, 2018; Stadie *et al.*, 2018; Boutilier *et al.*, 2020). Approaches in this category implicitly learn to explore, as they directly optimize the meta-RL objective whose maximization requires exploration. More specifically, the returns in the later $K - H$ episodes can only be maximized if the policy appropriately explores in the first K episodes, so maximizing the meta-RL objective can yield optimal exploration in principle. However, when more complicated exploration strategies are required, learning exploration this way can be extremely sample inefficient. One issue is that learning to exploit in the later $K - H$ episodes requires already having explored in the first K episodes, but exploration relies on good exploitation to provide reward signal (Liu *et al.*, 2021). For example, in the robot chef task, the robot can only learn to cook (i.e., exploit) when it has already found

Table 3.4: Few-shot meta-RL research categorized by exploration method as described in Section 3.4. End-to-End methods learn to explore implicitly, by directly maximizing the meta-RL objective. Posterior sampling maintains a distribution over possible tasks and acts optimally with respect to samples from this distribution. Task inference guides exploration in order to enable better inference of the task. Meta-exploration concerns exploration in the outer-loop.

Sub-topic	Papers
End-to-End Components	Stadie <i>et al.</i> (2018), García and Thomas (2019), and Boutilier <i>et al.</i> (2020)
Posterior Sampling	Gupta <i>et al.</i> (2018b), Rakelly <i>et al.</i> (2019), Kveton <i>et al.</i> (2021), and Simchowitz <i>et al.</i> (2021)
Task Inference	Zhou <i>et al.</i> (2019), Gurumurthy <i>et al.</i> (2020), Wang <i>et al.</i> (2020a), Liu <i>et al.</i> (2021), Fu <i>et al.</i> (2021), and Zhang <i>et al.</i> (2021)
Meta-Exploration	Zintgraf <i>et al.</i> (2021c) and Grewal <i>et al.</i> (2021)

all of the ingredients, but it is only incentivized to find the ingredients (i.e., explore) if doing so results in a cooked meal. Hence, it is challenging to learn to exploit without already having learned to explore and vice-versa, and consequently, end-to-end methods may struggle to learn tasks requiring sophisticated exploration in a sample-efficient manner, compared to methods with more structure, discussed later in this section.

One method that learns exploration end-to-end also modifies the reward to encourage exploration. E-RL² (Stadie *et al.*, 2018) is an end-to-end method that sets all rewards in the first K episodes to zero in the outer-loop. While ignoring these rewards does introduce sparsity, it may be helpful when myopically maximizing an immediate, dense reward prevents exploration needed for a longer-term reward. In general, many methods add more structure over end-to-end optimization of the meta-RL objective in order to solve task distributions that demand complicated exploration behavior.

Posterior sampling To circumvent the challenge of implicitly learning to explore, Rakelly *et al.* (2019) propose to directly explore via posterior sampling, an extension of Thompson sampling (Thompson, 1933) to MDPs. The inner-loop of this method, PEARL, is described in Algo-

rithm 3. When the agent is placed in a new task, the general idea is to maintain a distribution over what the identity of the task is, and then to iteratively refine this distribution by interacting with the task until it roughly becomes a point mass on the true identity. Posterior sampling achieves this by sampling an estimate of the task identity from the distribution on each episode, and acting as if the estimated task identity were the true task identity for the episode. Then, the observations from the episode are used to update the distribution either with black box methods (Rakelly *et al.*, 2019) or directly via gradient descent (Gupta *et al.*, 2018b).

Algorithm 3 PEARL Inner-Loop

- 1: Sample task $\mathcal{M} \sim p(\mathcal{M})$
 - 2: Initialize empty meta-trajectory \mathcal{D}
 - 3: **for** episode $0, \dots, H - 1$ sampled from \mathcal{M} **do**
 - 4: Sample $z \sim q_\theta(z|\mathcal{D})$
 - 5: Roll out $\pi_\theta(a|s, z)$ to collect trajectory τ for this episode
 - 6: Update \mathcal{D} with episode data, τ
 - 7: **end for**
-

Posterior sampling does, however, also have a drawback. First, since the policy employed is always conditioned on a sampled task, all of the exploration in such a method is executed by a policy that assumes it knows the task it is in. This means that the same policy is used for both exploration and exploitation. This can lead to suboptimal exploration in terms of Equation 2.3. Consider a robot chef that has to find a stove along a curved kitchen counter. Optimally exploring, the chef walks along the perimeter of the counter until it finds the stove. If the chef must be reset to its initial position, e.g., to charge its battery at the end of an episode, then the robot resumes exploration where it last left off. In contrast, a chef using posterior sampling, in each episode, simply walks to a different point along the counter that it has not yet checked, repeating this process until it finds the stove.

This comparison is depicted in Figure 3.9. Other methods for exploration in meta-RL exist that add structure to exploration without this limitation.

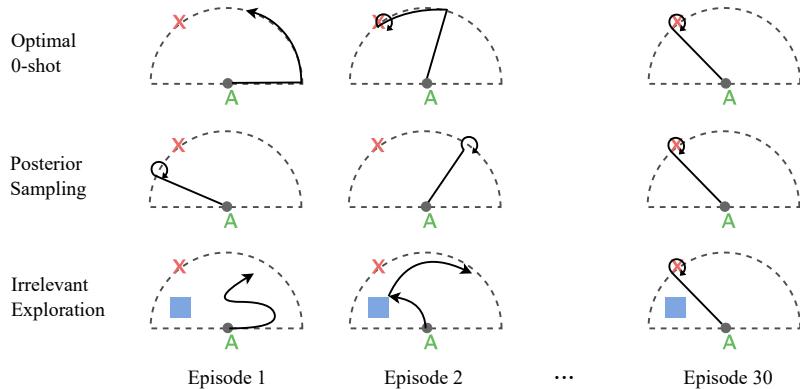


Figure 3.9: Exploration for a robot chef (green) finding a stove (red) along a circular counter. The first two rows compare optimal exploration and posterior sampling. The third row shows excessive exploration (blue) when irrelevant information is available in the sampled MDP.

Task inference Another way to avoid the challenges of implicitly learning to explore is to directly learn to explore using task inference objectives that encourage exploration (Zhou *et al.*, 2019; Gurumurthy *et al.*, 2020; Wang *et al.*, 2020a; Liu *et al.*, 2021; Fu *et al.*, 2021; Zhang *et al.*, 2021). Some, but not all, task inference methods make use of such an objective to encourage exploration. Exploration methods that use task inference generally add an intrinsic reward to gather information that removes uncertainty from the task distribution. In other words, these methods train the policy to explore states that enable predicting the task. Specifically, these intrinsic rewards generally incentivize improvement in transition predictions (i.e., in adapting the dynamics and reward function) (Zhou *et al.*, 2019) or incentivize information gain over the task distribution (Fu *et al.*, 2021; Liu *et al.*, 2021; Zhang *et al.*, 2021). The idea is that recovering the task is sufficient to learn the optimal policy, and hence achieve high returns in the later episodes. While task inference rewards may incentivize more exploration than necessary, as we discuss in this section in the context in meta-exploration, the rewards can be annealed so that the policy is ultimately optimized end-to-end (Zintgraf *et al.*, 2021c).

Most of these methods use separate policies for exploration and for exploitation, particularly when exploration episodes are freely available (Zhou *et al.*, 2019; Gurumurthy *et al.*, 2020; Liu *et al.*, 2021; Fu *et al.*, 2021). The intrinsic reward is used to train the exploration policy, while the standard meta-RL objective given by Equation 2.3 is used to train the exploitation policy. The exploration policy explores for the first K episodes, and then the exploitation policy exploits for the remaining $H - K$ exploitation methods, conditioned on the data collected by the exploration policy.

One method, DREAM (Liu *et al.*, 2021), first identifies the information that is useful to the exploitation policy, and then trains the exploration policy directly to recover only this information. This is critical when the number of shots K is too small to exhaustively explore all of the dynamics and reward function, much of which may be irrelevant. For example, exploring the decorations on the wall may provide information about the task dynamics, but are irrelevant for a robot chef trying to cook a meal. Learning the task representation in this way can be seen as multi-task training, as described in the Section 3.3, which addresses uninformative (e.g., a one-hot encoding), or irrelevant task information. This multi-task training is particularly beneficial in the context of exploration since the policy used in the multi-task phase to learn the task representation can also be reused as an exploitation policy. While, the exploration and exploitation policies could be meta-trained entirely in sequence, in practice they are trained simultaneously for stability (Liu *et al.*, 2021; Fu *et al.*, 2021). Meta-testing and meta-training for DREAM are described in Algorithms 4 and 5, respectively. Still, DREAM has some drawbacks. For instance, DREAM requires privileged information in the form of a known task representation, and DREAM may be suboptimal when it is not possibly to explore sufficiently in the given K shots (e.g., in a 0-shot setting).

Meta-exploration Finally, in meta-RL, there is still the process of acquiring data for the outer-loop learning, just as in standard RL. This is called *meta-exploration*, since it must explore the space of exploration strategies. While meta-exploration can be considered exploration in the outer-loop, both loops share data, and exploration methods may

Algorithm 4 DREAM Meta-Testing

-
- 1: Sample task $\mathcal{M} \sim p(\mathcal{M})$
 - 2: Initialize empty meta-trajectory \mathcal{D}
 - 3: **for** each exploration episode $k = 0, \dots, K$ sampled from \mathcal{M} **do**
 - 4: Roll out exploration policy $\pi_\theta(a|s, \mathcal{D})$ to collect trajectory τ
 - 5: Update \mathcal{D} with τ
 - 6: **end for**
 - 7: Infer $z \sim q_\theta(z|\mathcal{D})$
 - 8: **for** each exploitation episode $h = K, \dots, H - 1$ sampled from \mathcal{M} **do**
 - 9: Roll out exploitation policy $\pi_\theta^{\text{multi}}(a|s, z)$
 - 10: **end for**
-

affect both loops, so the distinction may be blurry. Often, sufficient meta-exploration occurs simply as a result of the exploration of the standard RL algorithm in the outer-loop. However, a common method to specifically address meta-exploration is the addition of an intrinsic reward. In fact, the addition of any task inference reward, discussed in the previous paragraph, can be considered meta-exploration. This is particularly apparent when considering that this intrinsic reward can be used to train a policy exclusively for off-policy data collection during meta-training. However, sometimes adding a task inference reward is not enough. In this case, intrinsic rewards can be added that function similarly to those in standard RL. For example, using random network distillation (RND) (Burda *et al.*, 2019), a reward may add an incentive for novelty (Zintgraf *et al.*, 2021c). In this case, the novelty is measured in the joint space of the state and task representation, instead of just in the state representation, as in standard RL. For example, HyperX (Zintgraf *et al.*, 2021c) adds the reward,

$$r^{\text{hyper}}(\phi, s) = \|f(\phi, s) - h(\phi, s)\|,$$

where $\phi = L_\theta(\mu, \sigma)$ represents a projection of the task distribution, as in VariBAD; f represents the predictor network in RND; and h represents the random prior network in RND.

Additionally, an intrinsic reward may add an incentive for getting data where the task inference has high error and is still not well trained

Algorithm 5 DREAM Meta-Training

```

1: while not done meta-training do
2:   Sample M tasks,  $\mathcal{M} \sim p(\mathcal{M})$ 
3:   for each task index  $i = 0, \dots, M$  do
4:     Initialize meta-parameters,  $\theta$ , and empty meta-trajectory  $\mathcal{D}^i$ 
5:     Sample  $z \sim g_\theta(z|c_{\mathcal{M}})$ , a task embedding with IB
6:     for each exploitation episode  $h = K, \dots, H - 1$  do
7:       Every other episode, replace true  $z$  with inferred  $\hat{z} \sim q_\theta$ ,
          to make the meta-training task distribution closer to the
          meta-test distribution for stability
8:       Roll out exploitation policy  $\pi_\theta^{\text{multi}}(a|s, z)$ 
9:       Update  $\pi_\theta^{\text{multi}}$  and  $g_\theta$  using task reward and prior
10:      end for
11:      for each exploration episode  $k = 0, \dots, K - 1$  do
12:        Roll out exploration policy  $\pi_\theta(a|s, \mathcal{D}^i)$  to collect trajectory
            $\tau$ 
13:        Update  $\mathcal{D}^i$  with  $\tau$ 
14:        Update  $q_\theta(z|\mathcal{D}^i)$  to infer  $z$  using variational inference
15:        Compute  $r_t^{\text{explore}} \leftarrow -\|z - \hat{z}_t\|_2^2 + \|z - \hat{z}_{t-1}\|_2^2 - c$  for constant
            $c$ , inferred task  $\hat{z}$ , and all timesteps  $t$ 
16:        Update  $\pi_\theta$  using exploration reward,  $r_t^{\text{explore}}$ , i.e. the infor-
           mation gain in  $q$  for each transition
17:      end for
18:    end for
19:  end while

```

(Zintgraf *et al.*, 2021c), or add an incentive for getting data where TD-error is high (Grewal *et al.*, 2021). For example, HyperX (Zintgraf *et al.*, 2021c) adds the reward:

$$\begin{aligned}
r^{\text{error}}(\phi, s) &= -\log p_\theta(s'|s, a, \phi) - \log p_\theta(r|s, a, \phi) \\
&\propto \|s' - \hat{s}'\|_2^2 + \|r' - \hat{r}'\|_2^2
\end{aligned}$$

Many of these rewards incentivize behavior that should not occur at test time, and in any case, the additional reward changes the optimal policy as suggested by Equation 2.3. To address this, the reward bonus can be

annealed, letting the portion incentivizing meta-exploration go to zero (Zintgraf *et al.*, 2021c), so that learning is still ultimately optimized end-to-end, or meta-training could occur off-policy.

3.5 Bayes-adaptive Optimality

Our discussion so far reveals two key intuitions about exploration. First, exploration reduces the uncertainty about the dynamics and reward function of the current task. Crucially, it is not optimal to indiscriminately reduce all uncertainty. Instead, optimal exploration only reduces uncertainty that increases expected *future* returns, and does not reduce uncertainty over distracting or irrelevant parts of the state space. Second, there is a tension between exploration and exploitation: gathering information to decrease uncertainty and increase future returns can sacrifice more immediate returns. In these cases, it can be worthwhile for the agent to fall back on behaviors shared across all tasks, instead of adapting to the task, particularly when time for exploration is limited (Lange and Sprekeler, 2020). Therefore, maximizing the returns across a period of time requires carefully balancing information gathering via exploration and exploiting this information to achieve high returns. These raise an important question: What is an optimal exploration strategy? To answer this question, we next introduce the Bayes-adaptive Markov decision process (Duff and Barto, 2002; Ghavamzadeh *et al.*, 2015), a special type of MDP whose solution is a Bayes-optimal policy, which optimally trades off between exploration and exploitation. Then, we discuss practical methods for learning approximate Bayes-optimal policies, and analyze the behavior of algorithms introduced in the previous section from the perspective of Bayes-optimality.

Bayes-adaptive Markov decision processes. To determine the optimal exploration strategy, we need a framework to quantify the expected returns achieved by a policy when placed into an MDP with unknown dynamics and reward function. From a high level, the *Bayes-adaptive Markov decision process* (BAMDP), a special type of MDP, models exactly this: At each timestep, the BAMDP quantifies the current uncertainty about an MDP and returns next states and rewards based

on what happens in expectation under the uncertainty. Then, the policy that maximizes returns under the BAMDP maximizes returns when placed into an unknown MDP. Crucially, the dynamics of the BAMDP satisfy the Markov property by augmenting the states with the current uncertainty. In other words, the optimal exploration strategy explicitly conditions on the current uncertainty to determine when and what to explore and exploit.

More formally, the BAMDP characterizes the current uncertainty as a distribution over potential transition dynamics and reward functions based on the current observations. Intuitively, a peaky distribution that places most of its mass on only a few similar dynamics and reward functions encodes low uncertainty, while a flat distribution encodes high uncertainty, as there are many different dynamics and reward functions that the agent could be in. Specifically, the belief at the t^{th} timestep $b_t = p(r, p \mid \tau_{:t})$ is a posterior over dynamics p and reward functions r consistent with the observations $\tau_{:t} = (s_0, a_0, r_0, \dots, s_t)$ so far, and the initial belief b_0 is a prior $p(r, p)$. Then, the states of the BAMDP are hyperstates $s_t^+ = (s_t, b_t)$ composed of a state s_t and a belief b_t , which effectively augments the state with the current uncertainty. Conditioned on the hyperstate, the policy can satisfy the Markov property, making the hyperstate a sufficient statistic of the history seen by the agent. The hyperstate is a natural sufficient statistic for Meta-RL, both for its relevance in the BAMDP, and since it corresponds to the belief-state of the meta-RL POMDP described in Section 2.3. However, other sufficient statistics for meta-RL exist as well (Choshen and Tamar, 2023; Subramanian *et al.*, 2022).

As previously mentioned, the transition dynamics and reward function of the BAMDP are governed by what happens in expectation under the current uncertainty. Specifically, the BAMDP reward function is the expected rewards under the current belief:

$$R^+(s_t, b_t, a_t) = \mathbb{E}_{R \sim b_t} [R(s_t, a_t)].$$

The transition dynamics of the BAMDP are similar. The probability of ending up at a next state s_{t+1} is the expected probability of ending up at that state under the belief, and the next belief is updated according

to Bayes rule based on the next state and reward from the *underlying MDP* and not the BAMDP:

$$\begin{aligned} P^+(s_{t+1}, b_{t+1} \mid s_t, b_t, a_t) \\ = \mathbb{E}_{R, P \sim b_t} [P(s_{t+1} \mid s_t, a_t) \delta(b_{t+1} = p(R, P \mid \tau_{:t+1}))]. \end{aligned}$$

In other words, the BAMDP can be interpreted as interacting with an unknown MDP and maintaining the current uncertainty (i.e., belief). Taking an action a_t at timestep t yields a next state s_{t+1} and reward r_{t+1} from the MDP, which are used to update the belief b_{t+1} . The next state in the BAMDP is then $s_{t+1}^+ = (s_{t+1}, b_{t+1})$, but the BAMDP reward is the expected reward under the current belief $r_t^+ = R^+(s_t, b_t, a_t) = \mathbb{E}_{R \sim b_t} [R(s_t, a_t)]$.

The standard objective on a BAMDP is to maximize the expected rewards over some horizon of H timesteps:

$$\mathcal{J}(\pi) = \mathbb{E}_{b_0, \pi} \left[\sum_{t=0}^{H-1} R^+(s_t, b_t, a_t) \right]. \quad (3.1)$$

As H increases, the agent is incentivized to explore more, as there is more time to reap the benefits of finding higher reward solutions. Notably, this objective exactly corresponds to the standard meta-RL objective (Equation 2.3), where the number of shots K is set to 0.

Learning an approximate Bayes-optimal policy Directly computing Bayes-optimal policies requires planning through hyperstates. As the hyperstates include beliefs, which are distributions over dynamics and reward functions, this is generally intractable for all but the simplest problems. However, there are practical methods for learning approximately Bayes-optimal policies (Lee *et al.*, 2019; Humplik *et al.*, 2019; Zintgraf *et al.*, 2020; Arumugam and Singh, 2022). The main idea is to learn to approximate the belief and simultaneously learn a policy conditioned on the belief to maximize the BAMDP objective (Equation 3.1).

As a concrete example, variBAD (Zintgraf *et al.*, 2020) learns to approximate the belief with variational inference. Since directly maintaining a distribution over dynamics and reward functions is generally intractable, variBAD represents the approximate belief with a distribution

$b_t = p(m | \tau_{:t})$ over latent variables m . This distribution and the latent variables m can be learned by rolling out the policy to obtain trajectories $\tau = (s_0, a_0, r_0, \dots, s_H)$ and maximizing the likelihood of the observed dynamics and rewards $p(s_0, r_0, s_1, r_1, \dots, s_H | a_0, \dots, a_{H-1})$ via the evidence lower bound. Simultaneously, a policy $\pi(a_t | s_t, b_t = p(m | \tau_{:t}))$ is learned to maximize returns via standard RL.

Connections with other exploration methods While not all the methods described in Section 3.4 aim to learn Bayes-adaptive optimal policies, the framework of BAMDPs can still offer a helpful perspective on how these methods explore. We discuss several examples below.

First, black box meta-RL algorithms such as RL^2 learn a recurrent policy that not only conditions on the current state s_t , but on the history of observed states, actions, and rewards $\tau_{:t} = (s_0, a_0, r_0, \dots, s_t)$, which is typically processed through a recurrent neural network to create some hidden state h_t at each timestep. Notably, this history is sufficient for computing the belief state $b_t = p(r, p | \tau_{:t})$, and hence black box meta-RL algorithms can in principle learn Bayes-adaptive optimal policies by encoding the belief state in the hidden state h_t . Indeed, variBAD can be seen as adding an auxiliary loss to a black box meta-RL algorithm that encourages the hidden state to be predictive of the belief state, though practical implementations of these approaches differ, as variBAD typically does not backpropagate through its hidden state, whereas RL^2 does. However, in practice, black box meta-RL algorithms may struggle to efficiently learn Bayes-adaptive optimal policies in domains requiring exploration that is temporally-extended and qualitatively different from the exploitation behavior. Learning such sophisticated exploration is difficult due to challenges in end-to-end optimization.

While Team *et al.* (2023) demonstrate that it is possible to learn complicated exploration strategies end-to-end, doing so may also require the use of curriculum learning, distillation, and a large number of samples for meta-training (Team *et al.*, 2023). Moreover, the meta-exploration problem in their task distribution may not have presented a difficult meta-exploration problem as their agent “does not use the trial conditioning information it observes to adjust its behaviour” (Team *et al.*, 2023). End-to-end training on the meta-RL objective alone presents

a difficult optimization problem. Liu *et al.* (2021) highlight one such optimization challenge for black box meta-RL algorithms, where learning to explore and gather information is challenging without already having learned how to exploit this information, and vice-versa.

Second, many exploration methods discussed in the previous section consider the few-shot setting, where the agent is given a few “free” episodes to explore, and the objective is to maximize the returns on the subsequent episodes. Likewise, the BAMDP objective can be modified to include free exploration episodes by setting initial rewards to zero. Depending on the amount of free exploration, the optimal policies for the BAMDP can encourage fairly different exploration behaviors. For example, in the robot chef task, the optimal few-shot exploration may be to first exhaustively look through the drawers and pantry for the best culinary utensils and ingredients in the initial few shots before beginning to cook. In contrast, optimal zero-shot behavior may attempt to locate the utensils and ingredients while cooking (e.g., as a pot of water boils), as spending the upfront time may be too costly. This may result in using less suitable utensils or ingredients, though, especially when optimized at lower discount factors.

More generally, methods designed for the few-shot setting attempt to reduce the uncertainty in the belief state in the initial few free episodes, and then subsequently exploit the relative low uncertainty to achieve high returns. This contrasts behavior in the zero-shot setting, which may involve interleaving exploration and exploitation. For example, the posterior sampling exploration in PEARL maintains a posterior over the current task, which is equivalent to the belief state. Then, exploration occurs by sampling from this distribution and pretending the sampled task is the current task, and then updating the posterior based on the observations, which aims to collapse the belief state’s uncertainty. Similarly, by learning an exploration policy that gathers all the task-relevant information during the few free exploration episodes, DREAM also attempts to collapse the belief state to include only dynamics and rewards that share the same optimal exploitation policy.

3.6 Supervision

In this section, we discuss most of the different types of supervision considered in meta-RL. In the standard setting discussed so far, the meta-RL agent receives reward supervision in both the inner- and outer-loops of meta-training, as well as meta-testing. However, this might not always be the case. Many variations have been considered, from the unsupervised case (i.e., complete lack of rewards during meta-training or testing), to stronger forms of supervision (e.g., access to expert trajectories or other privileged information during meta-training and/or testing). Each of these presents a different problem setting, with unique methods, visualized in Table 3.5. Below, we discuss these settings, from those with the least supervision to those with the most.

Unsupervised meta-RL The first problem setting provides the least supervision: no reward information is available at meta-train time, but it is available at meta-test time (Gupta *et al.*, 2018a; Jabri *et al.*, 2019; Mutti *et al.*, 2022). For example, a robot chef may be meta-trained in a standardized kitchen and then sold to customers, who may each have their own reward functions for the chef. However, the company training the robot may not know the desires of the customers. In this case, it is difficult to design the reward functions for the distribution of MDPs needed for meta-training. It is difficult even to define a distribution under which we might expect the test tasks to have support. One solution is simply to create rewards that encourage maximally diverse trajectories in the environment. Then, it is likely that what an end user desires is similar to one of these trajectories and reward functions. Gupta *et al.* (2018a) and Jabri *et al.* (2019) attempt to learn a set of diverse reward functions by rewarding behaviors that are distinct from one another. In general, a set of tasks can be created using an off-the-shelf unsupervised RL method (Eysenbach *et al.*, 2019; Park *et al.*, 2024). After this set of tasks is created, meta-RL can easily be performed as normal.

For example, Gupta *et al.* (2018a) use the method Diversity is All You Need (DIAYN, Eysenbach *et al.*, 2019), to create this distribution over reward functions. Specifically, DIAYN first learns a latent variable,

Table 3.5: Problem settings by supervision available at meta-training and meta-testing time as categorized in Section 3.6. Most of the literature in few-shot meta-RL considers the problem setting with rewards provided at meta-train and meta-test time. There are three additional variations on this supervision in meta-RL addressed in this section. Meta-imitation learning is a related but separate problem, briefly covered in this survey.

	Meta-Train	Meta-Test	Papers
Standard Meta-RL	Rewards	Rewards	See, e.g., Table 3.1
Unsupervised Meta-RL	Unsupervised	Rewards	Gupta <i>et al.</i> (2018a), Jabri <i>et al.</i> (2019), and Mutti <i>et al.</i> (2022)
Meta-RL with Unsupervised Meta-Testing	Rewards (Sparse)	Unsupervised (Sparse)	Sung <i>et al.</i> (2017), Miconi <i>et al.</i> (2018), Yang <i>et al.</i> (2019), Yan <i>et al.</i> (2020), and Najarro and Risi (2020)
Meta-RL via Imitation	Supervised	Rewards	Sparse: Gupta <i>et al.</i> (2018b), Rakelly <i>et al.</i> (2019), Zhao <i>et al.</i> (2021), and Wan <i>et al.</i> (2022) Mendonca <i>et al.</i> (2019), Weihs <i>et al.</i> (2021), Sharaf and Daumé III (2021), and Nguyen <i>et al.</i> (2022)
Mixed Supervision	Rewards and/or Supervision	Rewards and/or Supervision	Zhou <i>et al.</i> (2020), Prat and Johns (2021), Dance <i>et al.</i> (2021), and Rengarajan <i>et al.</i> (2022)
Meta-Imitation Learning	Supervised	(Un)-supervised	Duan <i>et al.</i> (2017), Finn <i>et al.</i> (2017b), James <i>et al.</i> (2018), Yu <i>et al.</i> (2018), Xu <i>et al.</i> (2019), Dasari and Gupta (2020), Bonardi <i>et al.</i> (2020), Singh <i>et al.</i> (2020), Brown <i>et al.</i> (2020), Jang <i>et al.</i> (2021), Li <i>et al.</i> (2021b), and Chowdhery <i>et al.</i> (2022)

Z , to parameterize the reward function, along with a multi-task policy, $\pi^{\text{multi}}(a|s, z)$. DIAYN does so by maximizing the mutual information between the state and the latent variable, ensuring partitioning of the states, and the entropy of the policy:

$$\begin{aligned} J_\theta &= H(A|S, Z) + I(S, Z), \\ &= H(A|S, Z) + H(Z) - H(Z|S) \end{aligned}$$

In practice, this optimization amounts to using soft actor critic (Haarnoja *et al.*, 2018) to maximize $H(A|S, Z)$, while setting $r(s, z) = \log q_\theta(z|s) - \log p(z)$ to maximize the other terms in expectation (Eysenbach *et al.*, 2019). Here, q_θ , is a learned discriminator predicting the probability of the latent variable given the state, and $p(z)$ is a known latent variable distribution. Note that q_θ can be learned by maximizing the same objective, J_θ , since optimizing the only term dependent on q_θ , $\log q_\theta(z|s)$, corresponds to a maximum likelihood estimate. After learning $q_\theta(z|s)$, this discriminator is then reused as a reward function for meta-RL. Specifically, a separate meta-RL agent (MAML) is trained over this latent distribution using $r(s, z) = \log q_\theta(z|s)$.

Once trained, such meta-RL agents can adapt more quickly than RL from scratch and are competitive, on several navigation and locomotion tasks, with meta-training over a hand-designed training distribution. Still, these domains are simple enough that diverse trajectories cover the task space, and a gap remains between these domains and more realistic domains such as the robot chef. Methods from reward-free RL (Touati and Ollivier, 2021), are highly relevant here, since they can also make use of access to the dynamics alone to learn representations. However, in that setting, the reward function is given to the agent at test-time, or estimated manually, rather than being inferred by a meta-learned algorithm.

Meta-RL with unsupervised meta-testing A second setting assumes rewards are available at meta-train time but none are available at meta-test time. For example, perhaps the company producing a robot chef is able to install many expensive sensors in several kitchens in a lab for meta-training. These sensors detect when a counter is scratched, or water damage occurs, or furniture is broken. All of these collectively are used to define the reward function. However, it may be prohibitively expensive to install these sensors in the house of each customer. In this case, rewards are not available at meta-test time. Rewards are in the outer-loop, but they are never used in the inner-loop, and it is assumed

that reward information is not needed to identify the tasks. In fact, only the dynamics vary across tasks in this setting.

In order to learn without rewards at meta-test time, many methods remove rewards from the inner-loop entirely. While the inner-loop cannot condition on reward, it may learn to maximize reward by maximizing its correlates. While black box methods are applicable in this setting off the shelf, assuming other inputs to the inner-loop correlate with reward, specific methods have been investigated. For example, in PPG normally the inner-loop requires sampling returns; however, without rewards, this is not possible at meta-test time. One solution is use a learned estimate of return, conditional on the data collected so far, to replace these returns in the policy gradient estimate. Here, the returns can be replaced with a learned advantage function, $A_\theta(s_t, a_t, s_{t+1})$ (Yang *et al.*, 2019) or learned critic $Q_\theta(s_t, a_t, \mathcal{D})$ (Sung *et al.*, 2017). For example, No-Reward Meta Learning (NoRML) modifies the MAML update to include a learned advantage function (in addition to offset parameters, ϕ_{offset} , that allow the meta-learned initialization to focus exclusively on exploration):

$$\phi_1^i = \phi_0 + \phi_{\text{offset}} + \alpha \mathbb{E}_{s_t, a_t, s_{t+1} \in \mathcal{D}_0^i} A_\theta(s_t, a_t, s_{t+1}) \nabla_{\phi_0} \log \pi_{\phi_0}(a_t | s_t).$$

Another approach is to leverage manually designed features in a black-box self-supervised inner-loop, while using a fully supervised outer-loop (Yan *et al.*, 2020). Finally, yet another approach is to leverage Hebbian learning (Hebb, 1949), a biologically inspired method for unsupervised learning in which weight updates are a function of the associated activations in the previous and next layers. The update to the weight ($w_{i,j}^k$) from the i th activation in layer k (x_i^k), to the j th activation in layer $k+1$ (x_j^{k+1}) generally has the form

$$w_{i,j}^k := w_{i,j}^k + \alpha(ax_i^k x_j^{k+1} + bx_i^k + cx_j^{k+1} + d),$$

where α is a learning rate and α, a, b, c, d are all meta-learned parameters in θ . Since weight updates only condition on adjacent layer activations, if no rewards are passed as input to the policy, this function is both local and unsupervised. Hebbian learning can be applied both to feed-forward (Najarro and Risi, 2020) and recurrent neural networks (Miconi *et al.*, 2018; Miconi *et al.*, 2019). Variants of Hebbian learning for

meta-RL, which pass rewards as an input to the policy are investigated by Chalvidal *et al.* (2022) and Rohani *et al.* (2022).

Alternatively, instead of having no rewards at meta-test time, we may have only sparse rewards. If dense rewards are available at meta-training, standard meta-RL methods can be applied directly by using the dense rewards in the outer-loop and sparse rewards in the inner-loop (Gupta *et al.*, 2018b; Rakelly *et al.*, 2019; Zhao *et al.*, 2021). In the case only sparse rewards are available for both meta-training and meta-testing, one approach is to alter the reward function at meta-training. A common method for this is a type of *experience relabelling* called *hindsight task relabelling* (Packer *et al.*, 2021; Wan *et al.*, 2022). Assuming tasks differ only in rewards, trajectories can be relabeled with rewards from other tasks and still be consistent with the MDP. Training can proceed by using a standard off-policy meta-RL algorithm (Rakelly *et al.*, 2019). This is particularly useful if, for instance, the trajectory did not reach a goal state in the original task, but did under the relabeled task. How to choose such a task is one area of investigation (Wan *et al.*, 2022). Alternatively, if the dynamics differ, one few-shot method allows for policies to be explicitly transferred between tasks, when helpful, by learning to map actions between tasks such that they produce similar state transitions in each task (Guo *et al.*, 2022). Yet another way of addressing sparse rewards at meta-test time is by introducing auxiliary rewards that encourage exploration, as discussed in Section 3.4.

Meta-RL via imitation A third setting assumes access to expert demonstrations at meta-training time, which provide more supervision than standard rewards. For example, a robot chef may have access to labeled supervision provided by human chefs. This setting can increase sample efficiency and reduce the burden of online data collection. This problem setting requires access to expert policies or expert data. If experts are not known in advance, they can be trained, per task. Alternatively, in the many-shot setting, policies optimized by existing RL algorithms can provide supervision, in a process known as algorithm distillation (Laskin *et al.*, 2023; Kirsch *et al.*, 2023). In order to improve over the existing RL algorithm, the learning speed can be increased by subsampling the demonstration data instead of learning on the full traces of expert data

(Kirsch *et al.*, 2023). One method, Guided Meta-Policy Search (GMPS, Mendonca *et al.*, 2019), proposes to imitate task experts in the outer-loop. In this case, the outer-loop can make use of supervised learning, while the inner-loop still learns a reinforcement learning algorithm that conditions on rewards at meta-test time. They specifically investigate the use of expert labels for the final policy of a MAML-style algorithm. If an expert is not available, the simultaneous training of task specific experts can also result in stable meta-training (Mendonca *et al.*, 2019).

GMPS illustrates meta-RL via imitation. GMPS rolls out an initial policy to compute a policy gradient for the inner-loop, and then uses supervised learning in the outer-loop for the adapted policy. The action labels for the supervised learning are given by expert policies for each task, which are assumed to be known. Here, the dataset aggregation entails using trajectories from the adapted policy but with expert actions labeled by the expert policies. This cloning loss can be written $J_{BC}(\mathcal{D}_{agg}^i, \pi_{\phi_1^i})$, where \mathcal{D}_{agg}^i is the aggregated dataset with expert actions for the i th task, $\pi_{\phi_1^i}$ is the policy adapted to the i th task after a single policy-gradient update, and J_{BC} is defined as

$$J_{BC}(\mathcal{D}, \pi) = \mathbb{E}_{s,a \sim \mathcal{D}}[\log \pi(a|s)]. \quad (3.2)$$

This method is described in Algorithm 6.

Meta-RL via imitation is still relatively unexplored. This may be due to difficulty in learning the correct supervision for exploratory actions. Obtaining a meta-RL expert requires knowing how to optimally explore in a meta-RL problem, which is generally hard to compute (Zintgraf *et al.*, 2020). Instead of obtaining such a general expert, these papers often make use of task-specific experts, which can be easily obtained by standard RL on each task. However, these task-specific experts can only provide supervision for the post-exploration behavior, e.g., for the actions taken by the final policy produced by the sequence of inner-loop adaptations in a MAML-style algorithm. In this case, credit assignment for the exploration policy may be neglected. Some papers use the same actions to provide supervision for both uninformed policies that must explore and informed policies that must exploit (Zhou *et al.*, 2020; Prat and Johns, 2021); however, in most environments actions are generally not the same for both exploration and exploitation. To get around

Algorithm 6 GMPS Meta-Training

```

1: Initialize meta-parameters  $\phi_0 = \theta$ 
2: while not done do
3:   Sample M tasks,  $\mathcal{M} \sim p(\mathcal{M})$ 
4:   Initialize outer-loop datasets,  $\mathcal{D}_{\text{agg}}^i$ , with states and actions from
      roll-outs of multi-task experts  $\pi^{\text{multi}}(a|s, \mathcal{M}^i)$ , for each task  $\mathcal{M}^i$ 
5:   for each task index,  $i = 0, \dots, M$  do
6:     Collect data  $\mathcal{D}_0^i$  using the initial policy  $\pi_{\phi_0}$ 
7:     Adapt policy parameters using a policy gradient step:  $\phi_1^i \leftarrow \phi_0 + \alpha \nabla_{\phi_0} \hat{J}_{RL}(\mathcal{D}_0^i, \pi_{\phi_0})$ 
8:   end for
9:   Update  $\phi_0$  using the meta-gradient:  $\phi_0 \leftarrow \phi_0 + \beta \nabla_{\phi_0} \frac{1}{M} \sum_i J_{\text{BC}}(\mathcal{D}_{\text{agg}}^i, \pi_{\phi_1^i})$ 
10:  Update  $\mathcal{D}_{\text{agg}}^i$  by adding states from roll-outs of adapted meta-RL
      agents,  $\pi_{\phi_1^i}$ , but actions from  $\pi^{\text{multi}}$ , for each task index  $i$ 
11: end while

```

this, the agent can adaptively switch between optimizing the meta-RL objective end-to-end and cloning the informed multi-task expert into (Weihs *et al.*, 2021). Additionally, it may be possible to use the multi-task policy to generate a reward encouraging similar state-action distributions between the multi-task policy and the meta-RL policy, particularly in maximum-entropy RL (Nguyen *et al.*, 2022).

Meta-imitation learning A fourth commonly studied problem setting is meta-imitation learning (meta-IL) (Duan *et al.*, 2017; Finn *et al.*, 2017b; James *et al.*, 2018; Yu *et al.*, 2018; Paine *et al.*, 2018; Seyed Ghasemipour *et al.*, 2019; Yu *et al.*, 2019; Goo and Niekum, 2019; Dasari and Gupta, 2020; Bonardi *et al.*, 2020; Singh *et al.*, 2020; Jang *et al.*, 2021; Li *et al.*, 2021a; Gao *et al.*, 2022). While meta-imitation learning is technically not meta-RL, because the inner-loop is not an RL algorithm, it is a closely related problem setting. Although an extensive survey of meta-IL methods is out of scope for this work, we briefly cover meta-IL here. This setting assumes access to a fixed set of demonstrations for each task in the inner-loop. Most methods additionally train the outer-

loop through behavioral cloning on fixed data, in a process also called meta-behavioral cloning (meta-BC). Alternatively, the outer-loop may also perform inverse RL, which we call meta-IRL (Seyed Ghasemipour *et al.*, 2019; Yu *et al.*, 2019; Xu *et al.*, 2019; Wu *et al.*, 2020; Hejna III and Sadigh, 2022). Another approach for using meta-learning for IRL is to train success classifiers via few-shot learning (Xie *et al.*, 2018; Li *et al.*, 2021b). Meta-IRL is generally performed online and so often requires a simulated environment, in contrast to meta-behavioral cloning. For both BC and IRL, the inner- and outer-loops generally also assume access to expert-provided actions, but one line of work considers inner-loops that use only sequences of states visited by an expert, potentially a human, despite being deployed on robotic system (Finn *et al.*, 2017b; Yu *et al.*, 2018; Dasari and Gupta, 2020; Bonardi *et al.*, 2020; Jang *et al.*, 2021).

There are many similarities between meta-RL research and meta-IL research. In meta-IL, there exist analogues to black-box methods (Duan *et al.*, 2017; James *et al.*, 2018; Dasari and Gupta, 2020; Bonardi *et al.*, 2020), PPG methods (Finn *et al.*, 2017b; Yu *et al.*, 2018), and task-inference methods (Jang *et al.*, 2021), as well as sim-to-real methods (James *et al.*, 2018; Bonardi *et al.*, 2020). For example, to adapt MAML to meta-BC, the inner-loop and outer-loop are each computed with a behavioral cloning loss over offline data, instead of using policy gradients. An adaptation of MAML for meta-BC, similar to Finn *et al.* (2017b), is shown in Algorithm 7. Likewise, to adapt RL² to meta-BC, the RNN summarizes an offline dataset, instead of online data, while the outer-loop uses behavioral cloning. An adaptation of RL² for meta-BC is shown in Algorithm 8.

Many contemporary papers discuss the meta-IL problem setting under the name of *in-context* learning (Brown *et al.*, 2020). In-context learning generally refers to learning in the activations of a sequence model, and frequently implies the use of transformers (Raparthy *et al.*, 2023; Lee *et al.*, 2024). In-context learning is a phenomenon, and meta-learning is an explicit way to achieve it. Black box methods and task inference methods can be seen as two ways to learn to elicit in-context learning. In-context learning can be used both in reference to in-context IL and in-context RL (Lee *et al.*, 2024). Large language models in particular, having gained significant traction recently (Devlin

Algorithm 7 Meta-Training MAML for Meta-BC

Require: A dataset of expert demonstrations per task index, i ,
for adaptation $\mathcal{D}_{\text{inner}}^i$, and validation, $\mathcal{D}_{\text{outer}}^i$

- 1: Initialize meta-parameters $\phi_0 = \theta$
- 2: **while** not done **do**
- 3: Sample M tasks, $\mathcal{M} \sim p(\mathcal{M})$
- 4: **for** each task index $i = 0, \dots, M$ **do**
- 5: Adapt policy parameters using a step of behavioral cloning on
the offline inner-loop data: $\phi_1^i \leftarrow \phi_0 + \alpha \nabla_{\phi_0} J_{\text{BC}}(\mathcal{D}_{\text{inner}}^i, \pi_{\phi_0})$
- 6: **end for**
- 7: Update ϕ_0 using the meta-gradient: $\phi_0 \leftarrow \phi_0 + \beta \nabla_{\phi_0} \frac{1}{M} \sum_i J_{\text{BC}}(\mathcal{D}_{\text{outer}}^i, \pi_{\phi_1^i})$
- 8: **end while**

Algorithm 8 Meta-Training RL² for Meta-BC

Require: A dataset of expert demonstrations per task index, i ,
for adaptation $\mathcal{D}_{\text{inner}}^i$ of length T, and validation, $\mathcal{D}_{\text{outer}}^i$

- 1: Initialize meta-parameters θ (RNN and other neural network pa-
rameters)
- 2: **while** not done **do**
- 3: Sample M tasks, $\mathcal{M} \sim p(\mathcal{M})$
- 4: **for** each task index $i = 0, \dots, M$ **do**
- 5: Initialize RNN hidden state as ϕ_0
- 6: Adapt task parameters using RNN on offline inner-loop dataset:
 $\phi_T^i \leftarrow f_\theta(\mathcal{D}_{\text{inner}}^i)$
- 7: **end for**
- 8: Update θ using the meta-BC objective: $\theta \leftarrow \theta + \beta \nabla_{\theta} \frac{1}{M} \sum_i J_{\text{BC}}(\mathcal{D}_{\text{outer}}^i, \pi_\theta(\cdot | \phi_T^i))$
- 9: **end while**

et al., 2019; Brown *et al.*, 2020; Chowdhery *et al.*, 2022), perform *in-context* learning (Brown *et al.*, 2020), which can be seen as explicit meta-IL where each prompt is treated as a task. Moreover, these models are capable of emergent meta-IL, i.e., meta-IL learning that is not explicitly in the training procedure, where novel datasets are passed

in text as a prompt to the model along with a query, e.g., in *few-shot prompting* (Chowdhery *et al.*, 2022), which suggests a similar capacity to be investigated in meta-RL.

Mixed supervision Beyond the most commonly studied settings above, Zhou *et al.* (2020), Prat and Johns (2021), Dance *et al.* (2021), and Rengarajan *et al.* (2022) consider a few related, but different settings. In these settings, there is a phase in which the inner-loop receives a demonstration, followed by a phase in which the inner-loop performs trial-and-error reinforcement learning. The demonstration generally comes from a fixed dataset collected offline, but may additionally be supplemented by online data from a separate policy trained via meta-IL on the fixed dataset (Zhou *et al.*, 2020). The demonstration data itself may (Prat and Johns, 2021) or may not (Dance *et al.*, 2021) provide supervision (actions and rewards), or a combination of the two (Zhou *et al.*, 2020). Finally, the agent may use reinforcement learning for supervision in the outer-loop during the trial-and-error phase (Dance *et al.*, 2021), it may use imitation learning (Zhou *et al.*, 2020), or it may use some combination of the two (Prat and Johns, 2021).

3.7 Model-based Meta-RL

So far, most of the algorithms discussed have been *model-free*, in that they do not learn a model of the MDP dynamics and reward. Alternatively, such models can be learned explicitly, and then be used for defining a policy via planning or be used for training a policy on the data generated by the model. Such an approach is called *model-based* RL, and can play a helpful role in meta-RL (Harrison *et al.*, 2018; Sæmundsson *et al.*, 2018; Nagabandi *et al.*, 2019b; Nagabandi *et al.*, 2019c; Galashov *et al.*, 2019; Mendonca *et al.*, 2020; Kaushik *et al.*, 2020; Lin *et al.*, 2020a; Lee *et al.*, 2020b; Perez *et al.*, 2020; Co-Reyes *et al.*, 2021a; Xian *et al.*, 2021; Hiraoka *et al.*, 2021; Seo *et al.*, 2022; Wang and Van Hoof, 2022; Shin *et al.*, 2022). Model-based meta-RL methods are summarized in Table 3.6. In this section, we briefly survey model-based meta-RL methods. We keep the discussion limited for several reasons: most model-based methods have an analogous model-free method al-

Table 3.6: Categories of model-based meta-RL methods from Section 3.7. Many themes are similar to those in the model-free section, such as the use of parameterized gradient descent and recurrent networks. The top half of the table shows key concepts in the representation of the inner-loop or the model adaptation, while the bottom half shows what the learned adaptive model is used for.

Sub-topic	Papers
Parameterized gradient descent	Nagabandi <i>et al.</i> (2019b), Mendonca <i>et al.</i> (2020), Kaushik <i>et al.</i> (2020), and Co-Reyes <i>et al.</i> (2021a)
Recurrent network	Nagabandi <i>et al.</i> (2019b) and Seo <i>et al.</i> (2022)
Variational inference	Sæmundsson <i>et al.</i> (2018) and Perez <i>et al.</i> (2020)
Model-predictive control	Nagabandi <i>et al.</i> (2019b), Lee <i>et al.</i> (2020b), and Shin <i>et al.</i> (2022)
Additional Data	Hiraoka <i>et al.</i> (2021), Rimon <i>et al.</i> (2022), Seo <i>et al.</i> (2022), and Shin <i>et al.</i> (2022)
Sub-procedure	Clavera <i>et al.</i> (2018) and Hiraoka <i>et al.</i> (2021)

ready discussed in Section 3, most trade-offs between model-based and model-free methods that exist in RL are the same in meta-RL, and most of the meta-RL literature considers model-free RL.

Many different types of model-based meta-RL have been investigated. In order to adapt the parameters of the model, some model-based meta-RL papers use gradient descent, as in MAML (Mendonca *et al.*, 2020; Kaushik *et al.*, 2020; Co-Reyes *et al.*, 2021a; Nagabandi *et al.*, 2019b), and some use black-box RNNs, as in RL2 (Nagabandi *et al.*, 2019b). Alternatively, a fixed number of past transitions can be encoded (Lee *et al.*, 2020b), or variational inference can be used (Sæmundsson *et al.*, 2018; Perez *et al.*, 2020). Similar themes arise in model-based meta-RL as in model-free meta-RL, such as the use of hypernetworks (Xian *et al.*, 2021) and permutation invariance of transitions (Galashov *et al.*, 2019; Wang and Van Hoof, 2022), as discussed earlier in Sections 3.2 and 3.3. Once the model of the environment is learned, it can be used to select the best sequence of actions by optimizing the sequence over a finite horizon. Generally, the first action is taken from this sequence, then the actions are re-planned. This process can be used to solve meta-RL with an off-the-shelf planner, and is called model-predictive

control (Nagabandi *et al.*, 2019b; Lee *et al.*, 2020b). Alternatively, some methods use the model simply for additional data when training a policy (Hiraoka *et al.*, 2021; Rimon *et al.*, 2022; Seo *et al.*, 2022), some uses model-based meta-RL as a sub-procedure in a standard RL algorithm to make learning more sample-efficient (Clavera *et al.*, 2018; Hiraoka *et al.*, 2021), and others use the adapted parameters from the learned model as an input to a policy that is then trained using standard RL (Mendonca *et al.*, 2020; Lee *et al.*, 2020b; Zintgraf *et al.*, 2020; Zhao *et al.*, 2021). However, in the last case there is significant overlap between model-based meta-RL and task-inference methods because learning to infer the task often means learning a latent-variable dynamics model. We discuss task-inference methods with model-free methods in Section 3.3, due to their otherwise similar assumptions.

A simple method, for example, would learn a black-box environment model, without a separate planner, and run a standard RL algorithm, but use the model for additional data in the policy gradient estimation. The model itself would be trained using a maximum likelihood estimate of the transition function and reward function over the same trajectories collected by the policy. For pseudocode of such a model, see Algorithm 9. On top of this, it is straightforward to condition the policy on parameters adapted for the transition or reward model, which can also be seen as a type of task inference method.

Overall, there are trade-offs between model-free meta-RL and model-based meta-RL. On one hand, model-based meta-RL methods can be extremely sample efficient when it is possible to learn an accurate model (Nagabandi *et al.*, 2019b). Model-based methods may also be learned off-policy, because the model can be trained using supervised learning. On the other hand, model-based meta-RL may require the implementation of additional components, especially for longer-horizon tasks that require more than an off-the-shelf planner, and can have lower asymptotic performance (Clavera *et al.*, 2018; Wu *et al.*, 2022). Finally, in the meta-learning context, model-based RL may confer two unique advantages. First, when there are too few tasks in the meta-training distribution, model-based meta-RL can allow for supplemental (imagined) tasks to be sampled (Rimon *et al.*, 2022). Second, model-based meta-RL may be easier when an off-the-shelf planner is viable

Algorithm 9 Black-Box Model-Based Meta-RL

- 1: Initialize an empty dataset for the environment model $\mathcal{D}_{\text{model}}$
- 2: Initialize an empty dataset for the policy $\mathcal{D}_{\text{policy}}$
- 3: Initialize meta-parameters, θ_m , for the model and meta-parameters, θ_p , policy
- 4: **while** not done **do**
- 5: Sample M tasks, $\mathcal{M} \sim p(\mathcal{M})$
- 6: **for** each task index $i = 0, \dots, M$ **do**
- 7: Replace old data in $\mathcal{D}_{\text{policy}}^i$ with roll-outs from π_ϕ using recurrent policy in real environment
- 8: Add the same roll-outs to $\mathcal{D}_{\text{model}}^i$
- 9: **end for**
- 10: Add roll-outs from π_ϕ in simulated environment model to $\mathcal{D}_{\text{policy}}$, optionally starting from real trajectories
- 11: Update policy θ_p using the meta-RL objective:

$$\theta_p \leftarrow \theta_p + \beta \nabla_{\theta_p} \frac{1}{M} \sum_i \hat{J}_{RL}(\mathcal{D}_{\text{policy}}^i, \pi_{\theta_p}(\cdot | \phi))$$
- 12: Update model θ_m using the supervised objective:

$$\theta_m \leftarrow \theta_m + \beta \nabla_{\theta_m} \frac{1}{M} \sum_i \mathbb{E}_{s,a,r,s',\phi \sim \mathcal{D}_{\text{model}}^i} [\log p_{\theta_m}(s'|s, a, \phi) + \log p_{\theta_m}(r|s, a, \phi)]$$
- 13: **end while**

and complicated exploration policies are necessary. In model-free meta-RL, the meta-learning needs to learn what data to collect and how. However, in model-based RL, the exploration may be offloaded to a planning algorithm. It may be easier to allow the planner to deal with complicated exploration in the inner-loop rather than learn this directly.

3.8 Theory of Meta-RL

The theory of meta-RL aims to understand and formalize the principles governing the learning of RL algorithms. As meta-RL is a relatively new area, its theoretical exploration is closely informed by insights from the related areas of meta-SL and multi-task representation learning for RL. However, meta-RL poses additional challenges due to the inner-loop learning an RL algorithm. For a theoretical understanding of these

challenges, many researchers have used the Bayesian framework. This Section integrates central insights from the theory of meta-SL and representation learning and overviews specific theoretical developments within meta-RL. The papers discussed in this section are summarized in Table 3.7.

Table 3.7: Categories of theory papers from Section 3.8. Papers relating to theory of meta-SL and representation learning for RL are included in addition to pure meta-RL theory papers.

Area	Papers
Meta-SL	Baxter (2000), Finn <i>et al.</i> (2019), Khodak <i>et al.</i> (2019), Denevi <i>et al.</i> (2019), Tripuraneni <i>et al.</i> (2021), and Collins <i>et al.</i> (2022)
Multi-task Representation Learning for RL	Jin <i>et al.</i> (2020), Hu <i>et al.</i> (2021), and Cheng <i>et al.</i> (2022)
Meta-RL	Simchowitz <i>et al.</i> (2021), Tamar <i>et al.</i> (2022), Rimon <i>et al.</i> (2022), Chen <i>et al.</i> (2022), and Ye <i>et al.</i> (2023)

Insights from supervised meta-learning Meta-learning, both meta-RL and meta-SL, focus on designing algorithms capable of efficiently learning new tasks from a given task distribution. A key aspect of this learning process is the development of representations or inductive biases that generalize across tasks. As an early theoretical contribution, Baxter (2000) laid the groundwork by studying the learning of inductive biases in a PAC (probably approximately correct, Valiant, 1984) setting, where the learner operates on a family related tasks and the goal is to find a hypothesis space that is appropriate for all of the tasks in the family. More recently, Tripuraneni *et al.* (2021) provide a fundamental result on how features learned across tasks improve the efficiency of learning in a new task. A related result specialized to MAML (Finn *et al.*, 2017a) is shown by Collins *et al.* (2022). MAML is further studied in the online convex optimization framework by Finn *et al.* (2019), Khodak *et al.* (2019), and Denevi *et al.* (2019), who assume a notion of task similarity under which all tasks are close to a single fixed point in the parameter space for which guarantees can be provided. These

findings, while not directly addressing meta-RL, offer valuable insights into the representation learning aspect crucial for meta-RL algorithms.

Multi-task representation learning for RL Multi-task representation learning for RL is closely related to meta-RL in that it considers learning a shared representation across a set of tasks. However, in representation learning, learning an exploration strategy or more generally learning a learning algorithm are not considered. Nevertheless, the settings are close enough that the theoretical results for multi-task representation learning can provide helpful guidance for developing a theoretical understanding of meta-RL.

Yang *et al.* (2020) and Hu *et al.* (2021) consider representation learning in linear MDPs. They show regret bounds across the set of parallel tasks that improve as the number of related tasks in the set increases. Additionally, Cheng *et al.* (2022) reveal that multitask representation learning in low-rank MDPs (Agarwal *et al.*, 2020) can significantly enhance sample efficiency when the total number of tasks surpasses a certain threshold. Their findings emphasize the efficiency of employing learned representations in downstream tasks, thereby illustrating the tangible benefits of multi-task representation learning in reinforcement learning.

Meta-RL specific theoretical advances While research specifically investigating the theory of meta-RL is limited, some important results have been obtained. In particular, results relating to the generalization bounds of meta-RL algorithms offer a principled motivation on the division of meta-RL methods into few-shot and many-shot methods introduced in Section 2.

Simchowitz *et al.* (2021), Tamar *et al.* (2022), and Rimon *et al.* (2022) focus on generalization in meta-RL. Tamar *et al.* (2022) provide PAC bounds on the number of training tasks required for learning an approximately Bayes-optimal policy, i.e., a policy that optimizes Equation 2.3. Rimon *et al.* (2022) show that the bound depends exponentially on the degrees of freedom of the task distribution $p(\mathcal{M})$. These results give a principled explanation of why meta-RL methods work well for

narrow task distributions but not for broader ones. Whereas Tamar *et al.* (2022) and Rimon *et al.* (2022) focus on in-distribution generalization, Simchowitz *et al.* (2021) study the problem of out-of-distribution generalization by considering misspecified priors for Thompson sampling algorithms, which are often considered in meta-RL (Rakelly *et al.*, 2019).

Continuing from the Bayesian perspective, Chen *et al.* (2022) bound the regret of the Bayes-optimal policy compared to the optimal policy on *any* MDP instance possible under the prior. While Chen *et al.* (2022) focus on the worst-case regret, Ye *et al.* (2023) bound the expected regret. Furthermore, they propose a pre-training and fine-tuning algorithm based on policy elimination that achieves favorable regret.

Finally, there are ongoing theoretical investigations targeted at MAML-like methods (Finn *et al.*, 2017a). Fallah *et al.* (2021) derive the sample complexity of a variant of the MAML algorithm. Liu *et al.* (2022b) and Tang (2022) investigate the bias and variance of gradient estimators for MAML and the latter propose a new gradient estimator with a favorable bias-variance trade-off.

4

Many-shot Meta-RL

In this section, we consider the many-shot setting where we want to, for example, learn a loss function that is applied on new tasks for thousands of updates, rather than just a handful. In other words, the goal is to learn a general purpose RL algorithm, similar to those currently used in practice. This setting is discussed separately from the few-shot setting presented in Section 3 because in practice it considers different problems and methods, even though increasing the trial length does not change the setting in principle. On one hand, a prototypical few-shot meta-RL problem is goal navigation in MuJoCo environments (Todorov *et al.*, 2012). In this scenario, the agent adapts to the different rewards defined by the goal but operates within the same environment. On the other hand, a typical task in many-shot meta-RL might involve learning an objective function for a policy-gradient method for Atari (Bellemare *et al.*, 2013) games.

In the few-shot multi-task setting, an adaptive policy can successfully solve unseen tasks in a small number of episodes by making use of a systematic exploration strategy that exploits its knowledge of the task distribution. This strategy works well for narrow task distributions, e.g., changing the goal location in a navigation task. For more complex

task distributions with more tenuous relationships between the tasks, the few-shot methods tend to not work as well (Rimon *et al.*, 2022; Mendonca *et al.*, 2020). Rimon *et al.* (2022) studies the complexity of meta-RL in terms of the degrees of freedom of the task distribution. See Section 3.8 for more details. Meta-RL methods designed for these more complicated conditions often resemble standard RL algorithms, which are in principle capable of solving any task given enough interactions. The drawback of standard RL algorithms is that they require many samples to solve tasks. The meta-RL algorithms building on them aim to improve the sample efficiency by explicitly optimizing for a faster algorithm. This algorithm template is illustrated in Figure 4.1. We label this setting *multi-task many-shot meta-RL* and discuss it further in Section 4.1.

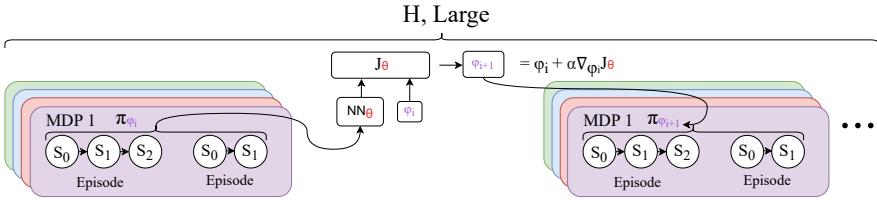


Figure 4.1: In many-shot meta-RL, the meta-parameters θ often parametrize a policy gradient objective function J_θ used for updating the parameters of the policy in the inner-loop ϕ . In this pattern, the inner-loop builds on the inductive bias of a policy gradient algorithm, which helps improve policies over long trials (large H). The meta-parameters are updated after a number of inner-loop updates, which may or may not correspond to the trial length depending on specific setting and the algorithm.

Another area where building on standard RL algorithms is helpful is *many-shot single-task meta-RL*. Instead of seeking greater generalization across complex task distributions, here the aim is to accelerate learning on a single hard task. Solving hard tasks requires many samples and leaves room for meta-learning to improve sample efficiency during the trial. We discuss this setting in Section 4.2. While the task distributions in the multi-task and single-task many-shot meta-RL problems are very different, we discuss the methods for both of them together in Section 4.3 due to the similarity of algorithms used in the settings.

There is less research on the many-shot meta-RL setting compared to its few-shot counterpart. We believe this disparity is in large part due to the higher computational demands that follow from the very nature of the many-shot meta-RL setting, placing it beyond the reach of many academic groups. Furthermore, optimization in the many-shot setting is challenging due to high inner-loop computational costs, which limit the kinds of inner-loops that can be efficiently trained. For example, RL^2 could technically be used in the many-shot setting, but optimizing an RNN over millions of timesteps is beyond the current state of the art. These challenges are open problems in the many-shot meta-RL setting, and are discussed further in Section 6.

4.1 Multi-task Many-shot Meta-RL

Many methods in this category are explicitly motivated by the desire to learn RL algorithms that learn quickly on *any* new task. This requires either a training task distribution that has support for all tasks of interest, or alternatively adaptation to tasks outside the training task distribution.

The objective for the many-shot multi-task setting is the same as in the few-shot setting and is given by Equation 2.3. The differences are the much longer length of the trial H and the broader task distribution, which may include differing action and observation spaces between the tasks. There is no strict cutoff separating long and short trial lengths but the algorithms targeting each setting are generally easy to distinguish. On common benchmarks, such as MuJoCo (Todorov *et al.*, 2012), algorithms targeting the few-shot setting reach maximum performance usually after at most tens of episodes, while algorithms for the many-shot setting are tested on benchmarks like Atari (Bellemare *et al.*, 2013) may take tens of thousands or more episodes to converge. In terms of environment steps, the inner-loop may require tens of thousands to tens of millions of interactions, depending on the environment (Oh *et al.*, 2020; Meier and Mujika, 2022). For the outer-loop to converge, this can require up to ten billion environment steps in total (Oh *et al.*, 2020; Jackson *et al.*, 2023). As in the few-shot setting, both meta-training and meta-testing consist of sampling tasks from a task distribution and running the inner-loop for H episodes.

In practice, since a single trial consists of many inner-loop updates, optimizing the meta-parameters over full trials can be challenging. Running a complete trial in the inner loop requires substantial computation by itself, which can make the meta-parameter updates too expensive to compute. Furthermore, even if it is feasible to sample full trials for each meta-parameter update, computing such updates may not be possible due to memory requirements or optimization problems like vanishing and exploding gradients.

For example, backpropagating through an agent’s entire lifetime necessitates storing all intermediate computations from each inner-loop update, leading to memory usage that grows linearly with the number of updates. In practice, this means that even processing a single batch can exhaust the memory capacity of standard devices like GPUs. Additionally, computing meta-gradients involves higher-order derivatives, increasing the computational complexity and often making it quadratic in the number of parameters.

Instead, the meta-learner is commonly updated to maximize a surrogate objective: the performance of the policy after a small number of inner-loop updates starting from the current parameters. This surrogate objective is a biased estimator of the policy performance at the end of the trial but can still provide gains in practice. The bias from considering a truncated horizon has been analyzed both in supervised learning (Wu *et al.*, 2018; Metz *et al.*, 2019; Metz *et al.*, 2021) and meta-RL (Vuorio *et al.*, 2021). We discuss mitigation strategies for the bias from truncation in Section 4.3.

A pseudocode template for a many-shot meta-RL algorithm is provided in Algorithm 10. In the algorithm, the trials may or may not finish depending on the number of iterations N between each outer-loop update. Updating the meta-parameters in the middle of a trial corresponds to the truncated objective described above.

In the multi-task many-shot setting, the meta-parameters are most commonly parameters of the objective function, which is differentiated with respect to the policy parameters. This results in a similar algorithm to MAML (Finn *et al.*, 2017a) described in Section 2.4. However, unlike in MAML, where the initialization of the policy is learned, the meta-parameters are in the update function, which enables consider-

Algorithm 10 Many-Shot Meta-Learning for Reinforcement Learning

```

1: Initialize meta-parameters  $\theta$ 
2: Sample  $M$  tasks  $\mathcal{M} \sim p(\mathcal{M})$ 
3: Initialize policy parameters  $\phi_0^i$  for each task  $\mathcal{M}^i$ 
4: Set  $j \leftarrow 0$ 
5: while not done do
6:   for  $N$  iterations do
7:     for each task index  $i = 0, \dots, M$  do
8:       Collect data  $\mathcal{D}_j^i$  using the policy  $\pi_{\phi_j^i}$ 
9:       Update policy parameters:  $\phi_{j+1}^i \leftarrow \phi_j^i + \alpha \nabla_{\phi_j^i} J_\theta(\mathcal{D}_j^i, \pi_{\phi_j^i})$ 
10:      end for
11:       $j \leftarrow j + 1$ 
12:    end for
13:    Update  $\theta$  to maximize the returns of the newest policies:  $\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{M} \sum_i J(\mathcal{D}_j^i, \pi_{\phi_j^i})$ 
14:    For tasks  $i$  for which the trial has concluded, re-initialize the
       policy parameters to  $\phi_0$ , and sample new tasks  $\mathcal{M} \sim p(\mathcal{M})$ .
15: end while

```

ing arbitrary policy parameterizations in the inner-loop. Furthermore, while learning the initialization is a general meta-parameterization, the parametrized update function can be easier to optimize, especially in the many-shot setting.

4.2 Single-task Many-shot Meta-RL

For some tasks in deep RL, useful training distributions of tasks may not be available. Moreover, even when they are, the individual tasks may require too many resources to solve on their own making multi-task training across them too hard. These tasks have motivated researchers to look at whether meta-RL can accelerate learning even when the task distribution consists of a single task, i.e., accelerating learning online during single-task RL training. Note that this setting is sometimes additionally referred to as *online cross-validation* (Sutton, 1992).

The goal in the single-task setting is to maximize the final performance of the policy after training has completed. The final performance is given by the meta-RL objective in Equation 2.3 when we choose a task distribution with only a single task, set the trial length H to a large number, and choose K such that only the last episode counts. Compared to the few-shot multi-task setting, when meta-learning on a single task, the optimization cannot wait until the inner-loop training has concluded and update only then because there is no further learning on which to use the updated meta-parameters. Therefore, single-task meta-learning necessarily follows a truncation pattern similar to the one described in Algorithm 10. In contrast to the many-shot multi-task meta-RL algorithm described in the pseudocode, in the single-task setting there is only one task and as a consequence only one set of policy parameters ϕ_j . Those parameters are usually never reset. Instead, the inner-loop keeps updating a single set of agent parameters throughout the lifetime of the agent. Therefore, Algorithm 10 is a fairly good representation of single-task meta-RL methods when choosing a task distribution with a single task and setting the trial length to the full length of training.

This meta-learning problem is inherently non-stationary, as the data distribution changes with the changing policy. The ability of the meta-learners to react to the non-stationary training conditions for the policy is often considered a benefit of using meta-learning to accelerate RL, but the non-stationarity itself results in a challenging meta-learning problem.

Many of the methods for single-task meta-RL are closely related to methods for online hyperparameter tuning. Indeed, there is no clear boundary between single-task meta-RL and online hyperparameter tuning, though generally hyperparameters refer to the special case where the meta-parameters θ are low-dimensional, e.g., corresponding to the learning rate, discount factor, or other hyperparameters of the RL algorithm. While such hyperparameters are often tuned by meta-RL algorithms (Xu *et al.*, 2018; Zahavy *et al.*, 2020), to limit the scope of our survey, we only consider methods that augment the standard RL algorithm in the inner-loop by introducing meta-learned components that do not have a direct counterpart in the non-meta-learning case. For a survey of methods for online hyperparameter tuning, see Parker-Holder *et al.* (2022).

4.3 Many-shot Meta-RL Methods

Algorithms for many-shot meta-RL aim to improve over the plain RL algorithms they build upon by introducing meta-learned components. The choice of meta-parameterization depends on the problem. The meta-parameterizations differ in how much structure they can capture from the task distribution, which matters in the multi-task case, and what aspects of the RL problem they address. The meta-parameterization may tackle problems such as credit assignment, representation learning, etc. The best choice of meta-parameterization depends on what aspect of the problem is the primary challenge.

Many of the topics discussed below, such as intrinsic rewards and hierarchical RL, are active research areas in RL on their own. Most of the research on these topics does not consider the bi-level structure present in meta-RL. We provide a concise description of each topic in general terms but do not provide details on the bodies of research outside of their intersection with meta-RL. For learning more about these topics, we provide references to foundational papers and surveys.

In the following, we discuss the different meta-parameterizations considered both in the single-task and multi-task settings. A summary of the methods discussed in this section is presented in Table 4.1, where the different methods are categorized by task distribution and meta-parameterization. The empty categories such as single-task meta-RL for learning hierarchical policies may be promising directions for future work. We also discuss the different outer-loop algorithms considered for many-shot meta-RL.

Learning intrinsic rewards The reward function of an MDP defines the task we want the agent to solve. However, the task-defining rewards may be challenging to learn from because maximizing them may not result in good exploratory behavior, e.g., when rewards are sparse (Singh *et al.*, 2009). One approach for making the RL problem easier is to introduce a new reward function that can guide the agent in learning how to explore. These additional rewards are called *intrinsic motivation* or *intrinsic rewards* (Aubret *et al.*, 2019). While intrinsic rewards are often designed manually, recently many-shot meta-RL methods have

Table 4.1: Many-shot meta-RL methods discussed in Section 4.3 categorized by the task distribution considered and meta-parameterization.

Meta-parameterization	Multi-task	Single-task
Intrinsic rewards	Zheng <i>et al.</i> (2020), Alet <i>et al.</i> (2020), Veeriah <i>et al.</i> (2021), Zou <i>et al.</i> (2021), and Meier and Mujika (2022)	Zheng <i>et al.</i> (2018) and Rajendran <i>et al.</i> (2020)
Auxiliary tasks		Veeriah <i>et al.</i> (2019), Lin <i>et al.</i> (2019), Zahavy <i>et al.</i> (2020), and Flennerhag <i>et al.</i> (2021)
Hyperparameter functions		Flennerhag <i>et al.</i> (2021), Almeida <i>et al.</i> (2021), Luketina <i>et al.</i> (2022), and Lu <i>et al.</i> (2022a)
Objective functions directly	Houthooft <i>et al.</i> (2018), Kirsch <i>et al.</i> (2019), Oh <i>et al.</i> (2020), Bechtle <i>et al.</i> (2021), and Jackson <i>et al.</i> (2023)	Xu <i>et al.</i> (2020)
Optimizers	Chen <i>et al.</i> (2017) and Lan <i>et al.</i> (2023)	
Black-box	Kirsch <i>et al.</i> (2022a)	

been developed to automate their design (Zheng *et al.*, 2018; Rajendran *et al.*, 2020; Veeriah *et al.*, 2021). One advantage meta-learning has over hand crafting the rewards is that it may be difficult to design reward functions that account for changes happening over time, but it is relatively easy to parameterize the learned rewards as functions of the entire trajectory seen so far (Zheng *et al.*, 2020; Alet *et al.*, 2020). Learning an intrinsic reward in the multi-task case can help the agent learn to explore the new environment more quickly (Zheng *et al.*, 2020; Alet *et al.*, 2020; Zou *et al.*, 2021). They can also help define skills as part of a hierarchical policy (Veeriah *et al.*, 2021) or be used as general reward shaping in the single-task case (Zheng *et al.*, 2018). Beyond the standard settings, Rajendran *et al.* (2020) learn intrinsic rewards for practicing in extrinsic reward-free episodes in-between evaluation episodes. Finally, Meier and Mujika (2022) present an unsupervised reward learning approach, which learns complex skills in Atari games.

Learning auxiliary tasks In some RL problems, learning a good representation of the observations is a significant challenge for which the RL objective alone may provide poor supervision. One approach for better representation learning is introducing *auxiliary tasks*, defined as unsupervised or self-supervised objectives optimized alongside the RL task (Jaderberg *et al.*, 2016). With auxiliary tasks the inner-loop objective then becomes

$$J_\theta(\mathcal{D}, \pi_\phi) = J^{\text{RL}}(\mathcal{D}, \pi_\phi) + J_\theta^{\text{aux}}(\mathcal{D}, \pi_\phi).$$

When a set of candidate auxiliary tasks is known, the best ones to use can be chosen by meta-learning the weight associated with each task, such that only the tasks that improve the outer-loop objective are assigned high weights (Lin *et al.*, 2019). Even when auxiliary tasks are not known in advance, meta-learning can be useful. Veeriah *et al.* (2019) show that the learned auxiliary tasks can improve sample efficiency over the base algorithm without auxiliary tasks and over handcrafted auxiliary tasks. The same approach for auxiliary task learning is used by Zahavy *et al.* (2020) and Flennerhag *et al.* (2021), who further improve the performance by tuning the hyperparameters of the inner-loop RL algorithm online. At the time of publication both achieved the state-of-the-art performance of model-free RL on the Atari benchmark (Bellemare *et al.*, 2013). For comparison of these methods with relevant baselines, see Table 4.2.

Table 4.2: Median human-normalized scores on the Arcade Learning Environment benchmark (Bellemare *et al.*, 2013) for many-shot meta-RL methods learning auxiliary tasks.

Method	Score (%)
Flennerhag <i>et al.</i> (2021)	611%
LASER (Not meta-RL) (Schmitt <i>et al.</i> , 2020)	431%
Zahavy <i>et al.</i> (2020)	364%
Xu <i>et al.</i> (2018)	287%
IMPALA (Not meta-RL) (Espeholt <i>et al.</i> , 2018)	192%

Learning functions that output hyperparameters Methods that learn functions that output hyperparameters of an inner-loop algorithm straddle the gap between hyperparameter optimization and meta-learning.

Flennerhag *et al.* (2021), Almeida *et al.* (2021), and Luketina *et al.* (2022) learn functions that take as inputs summary statistics of the inner-loop performance such as rewards and TD-error, and output the values of hyperparameters such as the λ -coefficient used in estimating returns. In temporal-difference learning, $\text{TD}(\lambda)$ is an algorithm that balances bias and variance by using the parameter λ to weight the contribution of bootstrapped multi-step returns, which introduce bias, and the Monte Carlo return estimates, which have high variance. Furthermore, Lu *et al.* (2022a) show that parameterizing the policy update size coefficient featured in algorithms such as proximal policy optimization (PPO) (Schulman *et al.*, 2017) by a meta-learned function can be beneficial. The parameters of these functions are themselves optimized by meta-gradients.

Modifying the RL objective directly Learning intrinsic rewards and auxiliary tasks shows that adding meta-learned terms to the RL objective can accelerate RL. These successes raise the question whether, instead of adding terms to the objectives, modifying the RL objectives directly via meta-RL can improve performance. To answer this question, Houthooft *et al.* (2018), Oh *et al.* (2020), and Xu *et al.* (2020) propose algorithms that replace the return or advantage estimator in a policy gradient algorithm with a learned function of the episode

$$\nabla_{\phi} J_{\theta}(\mathcal{D}, \pi_{\phi}) \propto \sum_{a_t, s_t \in \mathcal{D}} \nabla_{\phi} \log \pi_{\phi}(a_t | s_t) f_{\theta}(\mathcal{D}),$$

where $f_{\theta}(\mathcal{D})$ is some meta-learned function of the trajectory. An alternative to replacing the advantage estimator is proposed by Kirsch *et al.* (2019) and Bechtle *et al.* (2021), who consider a deep deterministic policy gradient (DDPG)-style (Lillicrap *et al.*, 2016) objective function, where the critic is learned via meta-RL instead of temporal difference (TD) learning.

$$\nabla_{\phi} J_{\theta}(\mathcal{D}, \pi_{\phi}) = \sum_{a_t, s_t \in \mathcal{D}} \nabla_{\phi} Q_{\theta}(s_t, \pi_{\phi}(a_t | s_t)),$$

where Q_{θ} is the meta-learned critic. Similar inner-loop is proposed for meta-IL by Yu *et al.* (2018). These learned RL objectives produce

promising results in both multi-task and single-task meta-RL. In the multi-task setting, Oh *et al.* (2020) demonstrate that an objective function learned on simple tasks such as gridworld can generalize to much more complicated tasks such as Atari (Bellemare *et al.*, 2013). Jackson *et al.* (2023) improve the generalization of the approach further by replacing the hand-designed training environments with an automated environment design component. Whereas in the single-task setting, Xu *et al.* (2020) show that the learned objective function can eventually outperform the standard RL algorithm (IMPALA, Espeholt *et al.*, 2018) it builds upon.

Learning Optimizers In most learning systems, the optimizer producing the parameter updates is manually designed. However, it is also possible to meta-learn an optimizer. Typically, the inner-loop of meta-learned optimizers conditions on losses and gradients, and outputs parameter updates. There has been some success in both many-shot and few-shot supervised learning to meta-learn the optimizer (Li and Malik, 2016; Andrychowicz *et al.*, 2016; Ravi and Larochelle, 2017). Some of these meta-learned optimizers use RL for meta-training (Li and Malik, 2016), and some deploy on MDPs (Chen *et al.*, 2017). Recently, a many-shot method has been proposed to meta-train and meta-test on MDPs, making it the first proper meta-RL method to learn an optimizer (Lan *et al.*, 2023). Since then, Goldie *et al.* (2024) have improved on this technique by additionally conditioning their optimizer on RL-specific heuristics.

Black-box meta-learning In few-shot meta-RL, black-box methods that use RNNs or other neural networks instead of stochastic gradient descent (SGD) tend to learn faster than the SGD-based alternatives. Kirsch *et al.* (2022a) argue that many black-box meta-RL approaches, e.g., Duan *et al.* (2016b) and Wang *et al.* (2016), cannot generalize well to unseen environments because they can easily overfit to the training environments. To combat overfitting, they introduce a specialized RNN architecture, which reuses the same RNN cell multiple times, making the RNN weights agnostic to the input and output dimensions and permutations. The proposed method requires longer trials to learn a

policy for a new environment, making it a many-shot meta-RL method, but in return it can generalize to completely unseen environments.

Outer-loop algorithms Regardless of the inner-loop parameterization chosen, by definition, algorithms for many-shot meta-RL have to meta-learn over long task-horizons. Directly optimizing over these long task horizons is challenging because it can result in vanishing or exploding gradients and has infeasible memory requirements (Sutskever, 2013; Metz *et al.*, 2021). Instead, as described above, most many-shot meta-RL algorithms adopt a surrogate objective, which considers only one or a few update steps in the inner-loop (Zheng *et al.*, 2018; Veeriah *et al.*, 2019; Kirsch *et al.*, 2019; Rajendran *et al.*, 2020; Zheng *et al.*, 2020; Zahavy *et al.*, 2020; Oh *et al.*, 2020; Veeriah *et al.*, 2021; Bechtle *et al.*, 2021). These algorithms use either A2C (Mnih *et al.*, 2016)-style (Zheng *et al.*, 2020; Oh *et al.*, 2020; Veeriah *et al.*, 2021; Bechtle *et al.*, 2021) or DDPG (Lillicrap *et al.*, 2016)-style (Kirsch *et al.*, 2019) actor-critic objectives in the outer-loop. Flennerhag *et al.* (2021) present a different kind of surrogate objective, which bootstraps target parameters for the inner-loop by computing several updates ahead and then optimizing its earlier parameters to minimize distance to that later target using a chosen metric. This allows optimizing over more inner-loop updates, and with the right choice of metric, it can be used for optimizing the behavior policy in the inner-loop, which is difficult using a standard actor-critic objective. This surrogate objective has also been used to meta-learning how to prioritize the task distribution for policy improvement in model-based methods (Burega *et al.*, 2022).

Alternatively *evolution strategies* (ES) (Rechenberg, 1971; Wierstra *et al.*, 2014; Salimans *et al.*, 2017), which are black-box optimization algorithms, are used by Houthooft *et al.* (2018), Kirsch *et al.* (2022a), and Lu *et al.* (2022a). ES works by sampling a set of parameters from a distribution, evaluating the set by running the inner-loop, and updating the parameters of the distribution. This can be seen as applying REINFORCE (Williams, 1992) on the parameter distribution. The general approach of using ES in the outer-loop is illustrated in Figure 4.2. ES suffers less from the vanishing and exploding gradients problem and has more favorable memory requirements at the cost of high variance

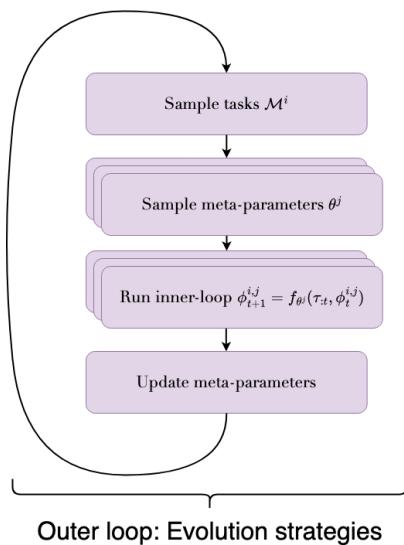


Figure 4.2: To estimate an update to the meta-parameters, ES samples a set of them and computes the inner-loop for each of them independently. This requires more evaluations of the inner-loop but can work better when the task-horizon is long.

and sample complexity compared to SGD-based methods (Metz *et al.*, 2021). Finally, genetic algorithms (Schmidhuber, 1987) and random search are used by Alet *et al.* (2020), Co-Reyes *et al.* (2021b), and Garau-Luis *et al.* (2022), who consider discrete parameterizations of the inner-loop objective.

5

Applications

In many application domains, fast adaptation to unseen situations during deployment is a critical consideration. By meta-learning on a set of related task, meta-RL provides a promising solution in these domains, such as traffic signal control (Zang *et al.*, 2020), building energy control (Luna Gutierrez and Leonetti, 2020; Grewal *et al.*, 2021), and automatic code grading in education (Liu *et al.*, 2022c). Meta-RL has also been used as a subroutine to address non-stationarity in the sub-field of continual RL (Al-Shedivat *et al.*, 2018b; Nagabandi *et al.*, 2019c; Riemer *et al.*, 2019; Berseth *et al.*, 2021; Liotet *et al.*, 2022; Woo *et al.*, 2022). In continual RL, the task, or MDP, can change throughout the lifetime of an agent. Meta-RL can help by learning how to address this non-stationarity in the task, given a distribution over sequences of tasks, simply by allowing the MDP to change to closely related MDPs, during a single trial. Moreover, curriculum learning and unsupervised environment design (UED) have been used to provide the distribution of tasks required for a meta-RL agent (Mehta *et al.*, 2020) or an otherwise adaptive agent (Dennis *et al.*, 2020). In this section, we consider where meta-RL has been most widely applied, as well as intersections with other sub-fields where meta-RL has been used successfully to solve problems. Specifically, we discuss robotics and multi-agent RL.

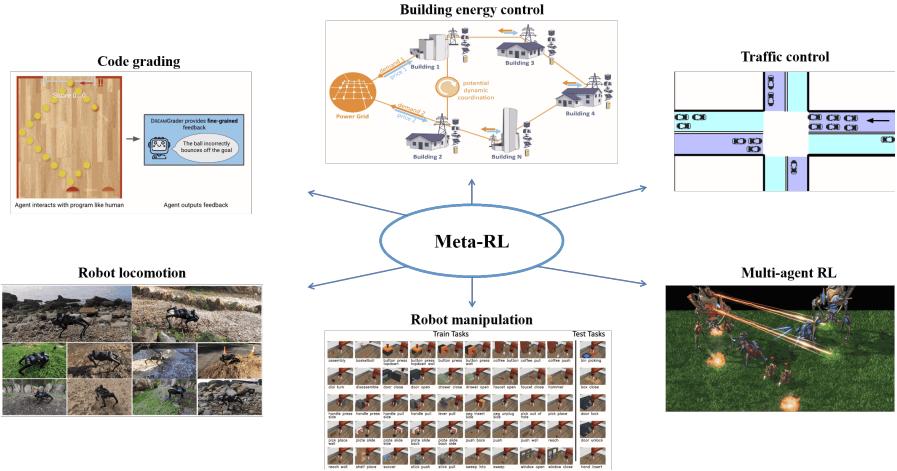


Figure 5.1: Example meta-RL application domains, including building energy control (Luna Gutierrez and Leonetti, 2020; Grewal *et al.*, 2021), traffic signal control (Zang *et al.*, 2020), multi-agent RL (Al-Shedivat *et al.*, 2018b; Grover *et al.*, 2018; Foerster *et al.*, 2018b; Papoudakis and Albrecht, 2020; Charakorn *et al.*, 2021; Zintgraf *et al.*, 2021a; Kim *et al.*, 2021; Feng *et al.*, 2021; Gupta *et al.*, 2021; Zhang and Chen, 2021; Yang *et al.*, 2022; Gerstgrasser and Parkes, 2022; He *et al.*, 2022; Lu *et al.*, 2022c), robot manipulation (Yu *et al.*, 2017; Akkaya *et al.*, 2019; Arndt *et al.*, 2020; Schoettler *et al.*, 2020; Ghadirzadeh *et al.*, 2021; Zhao *et al.*, 2022b), robot locomotion (Yu *et al.*, 2017; Schwartzwald and Papanikolopoulos, 2020; Yu *et al.*, 2020b; Song *et al.*, 2020b; Kumar *et al.*, 2021b; Kumar *et al.*, 2021a), and automatic code grading in education (Liu *et al.*, 2022c). Image credit to Vázquez-Canteli and Nagy (2019), Chen *et al.* (2020), Vázquez-Canteli *et al.* (2020), Yu *et al.* (2020a), Kumar *et al.* (2021a), Ellis *et al.* (2022), and Liu *et al.* (2022c).

5.1 Robotics

One important application domain of meta-RL is robotics, as a robot needs to quickly adapt to different tasks and environmental conditions during deployment, such as manipulating objects with different shapes, carrying loads with different weights, etc. Training a robot from scratch for each possible deployment task may be unrealistic, as RL training typically requires millions of steps, and collecting such a large amount of data on physical robots is time-consuming, and even dangerous when the robots make mistakes during learning. Meta-RL provides a promising solution to this challenge by meta-learning inductive biases from a set of related tasks to enable fast adaptation in a new task.

However, while meta-RL enables efficient adaptation during deployment, it comes at the expense of sample-inefficient meta-training on multiple tasks, which is bottlenecked by the cost of collecting an even larger amount of real-world data for meta-training. Nonetheless, some methods meta-train in the real world (Nagabandi *et al.*, 2019a; Belkhale *et al.*, 2021; Zhao *et al.*, 2021; Zhao *et al.*, 2022b; Walke *et al.*, 2022), and even learn to manually reset their tasks while doing so (Walke *et al.*, 2022). However, most methods train a meta-RL agent in simulation instead of on physical robots, where different tasks can be easily created by changing the simulator parameters, and then deploy the meta-trained robot in the real world. This process, known as *sim-to-real transfer* (Zhao *et al.*, 2020), significantly reduces the meta-training cost and is adopted by most methods that apply meta-RL to robotics (Yu *et al.*, 2017; Akkaya *et al.*, 2019; Arndt *et al.*, 2020; Cong *et al.*, 2020; Kaushik *et al.*, 2020; Schoettler *et al.*, 2020; Schwartzwald and Papanikolopoulos, 2020; Song *et al.*, 2020b; Yu *et al.*, 2020b; Ghadirzadeh *et al.*, 2021; Kumar *et al.*, 2021b; Kumar *et al.*, 2021a; He *et al.*, 2022). We give a taxonomy of these robotic meta-RL papers in Table 5.1 and introduce them in more detail below.

Model-free meta-RL methods directly adapt the control policy to handle unseen situations during deployment, such as locomotion of legged robots with different hardware (mass, motor voltages, etc.) and environmental conditions (floor friction, terrain, etc.) (Yu *et al.*, 2017; Schwartzwald and Papanikolopoulos, 2020; Yu *et al.*, 2020b; Song *et al.*, 2020b; Kumar *et al.*, 2021b; Kumar *et al.*, 2021a), and manipulation with different robot arms or variable objects (Yu *et al.*, 2017; Akkaya *et al.*, 2019; Arndt *et al.*, 2020; Schoettler *et al.*, 2020; Ghadirzadeh *et al.*, 2021; Zhao *et al.*, 2022b). On one hand, black-box methods (Akkaya *et al.*, 2019; Schwartzwald and Papanikolopoulos, 2020) (see Section 3.2) and task-inference methods (Yu *et al.*, 2017; Schoettler *et al.*, 2020; Yu *et al.*, 2020b; Kumar *et al.*, 2021b; Kumar *et al.*, 2021a; He *et al.*, 2022; Zhao *et al.*, 2022b) (see Section 3.3), when used in robotics, generally condition the policy on a context vector inferred from historical data to represent the current task. They mainly differ in what kind of loss function is used to train the task inference. On the other hand, model-free PPG methods (see Section 3.1) in robotics

Table 5.1: Taxonomy of the meta-RL papers that apply meta-RL to robotics from Section 5.1. Methods for meta-RL can be directly applied to robotics and robotics span the range of meta-RL methods. Task-inference and Model-based methods are common in this section for their sample efficiency. Real-world meta-training is relatively less common.

		Black-box	Task-inference	PPG
Model-free	Sim-to-real	Akkaya <i>et al.</i> (2019) and Schwartzwald and Panikolopoulos (2020)	Yu <i>et al.</i> (2017), Schoettler <i>et al.</i> (2020), Kumar <i>et al.</i> (2021b), Kumar <i>et al.</i> (2021a), and He <i>et al.</i> (2022)	Gao <i>et al.</i> (2019), Arndt <i>et al.</i> (2020), Song <i>et al.</i> (2020b), Yu <i>et al.</i> (2020b), and Ghadirzadeh <i>et al.</i> (2021)
	Real-world meta-training	-	Zhao <i>et al.</i> (2022b), Zhao <i>et al.</i> (2021), and Walke <i>et al.</i> (2022)	-
Model-based	Sim-to-real	Cong <i>et al.</i> (2020)	-	Kaushik <i>et al.</i> (2020) and Anne <i>et al.</i> (2021)
	Real-world meta-training	Nagabandi <i>et al.</i> (2019a) and Belkhale <i>et al.</i> (2021)	-	-

(Gao *et al.*, 2019; Arndt *et al.*, 2020; Song *et al.*, 2020b; Ghadirzadeh *et al.*, 2021) mainly build upon MAML (Finn *et al.*, 2017a). However, they usually require more rollouts for adaptation compared to model-free black-box methods, which is consistent with our discussion on the efficiency of different methods in Section 3. Moreover, instead of directly using MAML, these methods introduce further modifications to enable sample-efficient training (Arndt *et al.*, 2020; Ghadirzadeh *et al.*, 2021) and handle the high noise of the real world (Song *et al.*, 2020b).

Another line of works adopts model-based meta-RL (Section 3.7). Model-based methods may be more suitable for robotic tasks for the following reasons: (1) Model-based RL can be more efficient than model-free methods, a key consideration for robot deployment (Polydoros and Nalpantidis, 2017). (2) Adapting the dynamics model may be much easier than adapting the control policy in some cases, such as in tasks where task difference is defined by various dynamics parameters. Similar to the model-free case, both black-box methods (Nagabandi *et al.*, 2019a; Cong *et al.*, 2020; Belkhale *et al.*, 2021) and PPG methods

(Nagabandi *et al.*, 2019a; Kaushik *et al.*, 2020; Anne *et al.*, 2021) have been considered adapting the dynamics model for model-based control. (We generally avoid classification of methods as task-inference and model-based. See Section 3.7.)

While many meta-RL methods can be applied to robotics, the critical challenges in practice are often different from the conceptual problems studied in the field as a whole. For example, exploration, which differentiates meta-RL from supervised meta-learning, is not generally the most challenging obstacle in meta-RL for robotics. Often, it is the case that robots must generalize over a distribution over robot platforms or terrains, which present themselves immediately to the agent. Instead, the sim-to-real gap, induced by the simulator generally required for meta-training, presents a significant hurdle. Some meta-RL methods do mitigate the impact of sim-to-real transfer. For example, the task inference algorithm, IMPORT, discussed in Section 3.3, leverages privileged information in the simulator. A nearly identical algorithm has been used successfully in robotics sim-to-real transfer in the robotics literature (Kumar *et al.*, 2021a), despite not being framed as meta-RL in that context. This phenomenon demonstrates how familiarity with meta-RL could be of immense use to the field at large in practice in the future.

5.2 Multi-agent RL

Meta-learning has been applied in multi-agent RL to solve a number of problems, from learning with whom to communicate (Zhang and Chen, 2021), to automating mechanism design by learning agent-specific reward functions (Yang *et al.*, 2022). Notable problems and solutions are summarized in Table 5.2. However, in this section we focus on two main problems that meta-RL can address in the multi-agent setting. First, we introduce the problems of generalization to unseen agents and non-stationarity, and discuss how meta-RL can address them in general. Then, we discuss the types of meta-RL methods that have been used to address each problem, elaborating on PPG methods that propose additional mechanisms for each problem.

Table 5.2: Summary of meta-RL papers used for multi-agent RL from Section 5.2. Methods use PPG, black-box, and task-inference agents, in addition to non-adaptive agents. As examples, the problems addressed include generalization over other agents policies and non-stationarity induced by other agents.

Sub-topic	Papers
PPG meta-gradients for non-stationarity	Al-Shedivat <i>et al.</i> (2018b), Foerster <i>et al.</i> (2018b), and Kim <i>et al.</i> (2021)
Black box for generalization to teammates	Charakorn <i>et al.</i> (2021)
Black box for generalization to opponents	Lu <i>et al.</i> (2022c)
Task inference for generalization to teammates	Grover <i>et al.</i> (2018) and Zintgraf <i>et al.</i> (2021a)
Task inference for generalization to opponents	Grover <i>et al.</i> (2018), Papoudakis and Albrecht (2020), and Zintgraf <i>et al.</i> (2021a)
Task inference for generalization to humans	He <i>et al.</i> (2022)

The first multi-agent problem we consider is generalization over other agents. In multi-agent RL, many agents act in a shared environment. Often, it is the case that other agents' policies vary greatly. This creates a problem of generalization to unseen agents. This generalization may occur over opponents (Papoudakis and Albrecht, 2020; Lu *et al.*, 2022c), or over teammates (Charakorn *et al.*, 2021; Gupta *et al.*, 2021; He *et al.*, 2022), which is sometimes called ad hoc teamwork (Stone *et al.*, 2010). The other agents may be learned policies (Zintgraf *et al.*, 2021a) or even humans (He *et al.*, 2022). By viewing other agents as (part of) the task, and assuming a distribution of agents available for practice, meta-RL is directly applicable. This view can lead to the application of meta-learning methods (Beck *et al.*, 2022; Zintgraf *et al.*, 2020) to the multi-agent setting (Zintgraf *et al.*, 2021a; Tessera *et al.*, 2024). Using meta-RL to address generalization over other agents is visualized in Figure 5.2.

The second multi-agent problem we consider is *non-stationarity*. In multi-agent RL, from the perspective of any one agent, all other agents change as they learn. This means that from one agent's perspective, if all of the other agents are modeled as part of the environment, then the

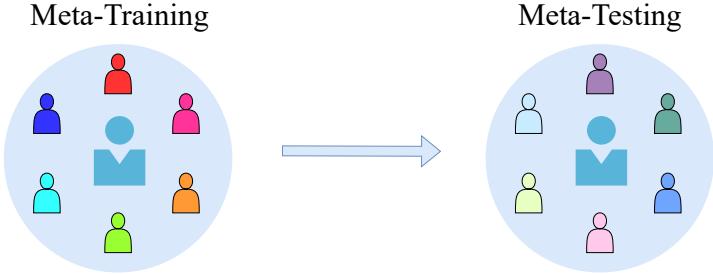


Figure 5.2: Illustration of using meta-RL to address generalization over unseen agents. By generalizing over other agents at meta-train time, we can adapt to new agents at meta-test time.

environment changes – i.e., the problem is non-stationary (Al-Shedivat *et al.*, 2018b; Foerster *et al.*, 2018b; Kim *et al.*, 2021). Meta-RL likewise can address non-stationary by treating other learning agents as (part of) the task. In this case, the learning algorithm of each agent, and what each agent has learned so far, collectively defines the task. By repeatedly resetting the other learning agents during meta-training, we can meta-learn how to handle the changes introduced by the other agents. From the perspective of the meta-learning agent, the distribution over other agents remains stationary. This effectively resolves the non-stationarity of multi-agent RL, which is depicted in Figure 5.3.

Solutions in multi-agent RL make use of all different types of meta-RL methods: PPG methods (Al-Shedivat *et al.*, 2018b; Foerster *et al.*, 2018b; Kim *et al.*, 2021), black-box (Charakorn *et al.*, 2021; Lu *et al.*, 2022c), and task-inference methods (Grover *et al.*, 2018; Papoudakis and Albrecht, 2020; Zintgraf *et al.*, 2021a; He *et al.*, 2022). These methods are discussed in Sections 3.1, 3.2, and 3.3, respectively. The agent may even use a Markovian (i.e., non-adaptive) policy, if the other agents are learning (Lu *et al.*, 2022b). Most of these methods can be applied without modification to the underlying meta-RL problem to address both generalization over other agents and non-stationarity. In the case that other agents form the entirety of the task, then the meta-RL objective can be written:

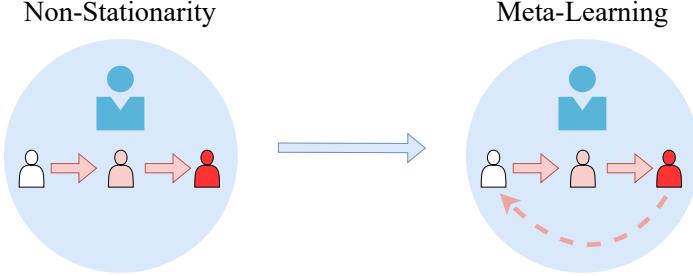


Figure 5.3: Illustration of using meta-RL to train over another agent’s learning in multi-agent RL. By resetting the learning process of other agents, and training over it, we can learn to address the non-stationarity created by the learning of other agents.

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi'_i \sim p(\pi'_i) \forall i=0 \dots A} \left[\mathbb{E}_{\mathcal{D}} \left[G(\mathcal{D}) \middle| \pi_{f_\theta}, \pi'_0 \dots \pi'_A \right] \right], \quad (5.1)$$

where A is the number of other agents. However, several PPG papers investigate additional mechanisms for both such generalization and non-stationarity that we discuss next.

Some PPG papers focus on improving the distribution of other agents using meta-gradients, in order to improve generalization to new agents. See Section 3.1 for a discussion of meta-gradient estimation. These papers focus on the curriculum of opponents to best support learning in a population-based training setting. Gupta *et al.* (2021) iterates between PPG meta-learning over a distribution of fixed opponents and adding the best-response to the meta-learned agent back to the population. Alternatively, Feng *et al.* (2021) uses meta-gradients or evolutionary strategies to optimize a neural network to produce opponent parameters. The opponent parameters are chosen to maximize the learning agent’s worst case performance. Both methods use the distribution of agents to create a robust agent capable of generalizing across many other agents.

Finally, some PPG papers introduce additional mechanisms to handle non-stationarity. In order to resolve all non-stationarity, all other (adaptive or non-adaptive) agents must repeatedly reset to their initial policies eventually. While this has been explored (Papoudakis and Albrecht, 2020; Zintgraf *et al.*, 2021a; Charakorn *et al.*, 2021), PPG methods tend to allow other agents to continue to learn (Al-Shedivat

et al., 2018b; Foerster *et al.*, 2018b), or even to meta-learn (Kim *et al.*, 2021), without resetting. Each PPG method addresses the non-stationary learning of other agents in a different way. For example, Al-Shedivat *et al.* (2018b) propose meta-learning how to make gradient updates such that performance improves against the subsequent opponent policy, over various pairs of opponent policies. In contrast, Foerster *et al.* (2018b) derive a policy gradient update such that one agent meta-learns assuming the rest follow an exact policy gradient, and Kim *et al.* (2021) derive a policy gradient update assuming all agents sample data for meta-learning.

6

Open Problems

Meta-RL is an active area of research with an increasing number of applications, and in this nascent field there are many opportunities for future work. In this section, we discuss some of the key open questions in meta-RL. Following the method categories in this survey, we first discuss some important directions for future work in few-shot and many-shot meta-RL in Section 6.1 and 6.2 respectively. Following this, we discuss how to utilize offline data in Section 6.3 in order to eliminate the need for expensive online data collection during adaptation and meta-training. Finally, we take a critical stance and evaluate some potential limitations of meta-RL research.

6.1 Few-shot Meta-RL Generalization

As discussed in Section 3, few-shot meta-RL methods are a promising solution to fast adaptation on new tasks. However, so far their success is mainly achieved on narrow task distributions, while the ultimate goal of meta-RL is to enable fast acquisition of entirely new behaviors. Consequently, future work should focus more on generalization of few-shot meta-RL methods to broader task distributions. A straightforward way to achieve this goal is to meta-train on a broader task distribution

to learn an inductive bias that can generalize to more tasks. However, training on a broader task distribution also introduces new challenges (such as a harder exploration problem) that are beyond the scope of existing methods. For an overview of methods for generalization in RL, beyond meta-RL, see Kirk *et al.* (2023). Moreover, even when trained on a broad task distribution, the agent may still encounter test tasks that lie outside the training distribution. Generalization to such out-of-distribution (OOD) tasks is important since a meta-RL agent is likely to encounter unexpected tasks in the real world. For a summary of the task distributions in existing benchmarks, see Table 6.1. We discuss these two open problems, illustrated in Figure 6.1, in more detail below.

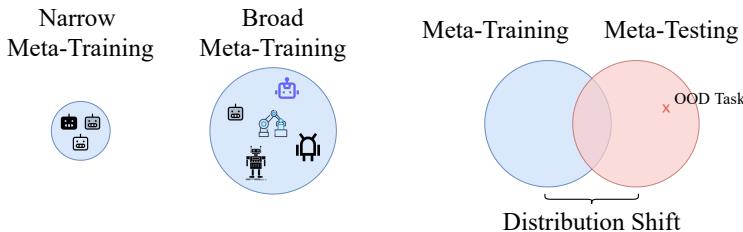


Figure 6.1: Illustration of broad task distributions (left) and OOD generalization (right). There is a clear need for meta-training distributions with both more tasks and more diverse tasks. Novel methods may be needed for such distributions, including methods that fail gracefully when out of distribution.

Training on broader task distributions for better generalization The meta-training task distribution plays an important role in meta-RL, as it determines *what* inductive bias we can learn from data, while the meta-RL algorithms determine *how well* we can learn this inductive bias. The task distribution should be both diverse enough so that the learned inductive bias can generalize to a wide range of new tasks, and clear enough in task structure so that there is indeed some shared knowledge we can utilize for fast adaptation.

However, existing few-shot meta-RL methods are frequently meta-trained on narrow task distributions, where different tasks are simply defined by varying a few parameters that specify the reward function or environment dynamics (Duan *et al.*, 2016b; Finn *et al.*, 2017a; Rakelly

Table 6.1: Summary of existing benchmarks in meta-RL by setting and number of tasks. MuJoCo and 2D Navigation are perhaps the most common, but also the narrowest meta-training distributions. XLand, Alchemy, NetHack, and Procgen are procedurally generated and so contain many tasks. NetHack and Procgen, however, were not designed to evaluate adaptation in meta-RL, and XLand is private. Additionally, benchmarks that contain both discrete tasks and diverse behavior generally require many tasks in order to generalize to new tasks.

Benchmark Name	Citation	Setting	Number of Tasks
XLand 2.0 (private)	Team <i>et al.</i> (2023)	Game (Diverse)	$> 10^{40}$
Alchemy	Wang <i>et al.</i> (2021)	Game (Diverse)	$> 167,424$
NetHack	Küttler <i>et al.</i> (2020)	Game (Diverse)	$>> 200,000$
Procgen	Cobbe <i>et al.</i> (2020)	Game (Diverse)	$>> 200,000$
Sonic	Nichol <i>et al.</i> (2018b)	Game (Diverse)	58 discrete
Meta Arcade	Staley <i>et al.</i> (2021)	Game (Diverse)	24; continuous (e.g., ball size)
DreamerGrader	Wang <i>et al.</i> (2021) e.g., Finn <i>et al.</i> (2017a) and Zintgraf <i>et al.</i> (2020)	Game (Narrow)	3,556 discrete
2D Navigation	Zintgraf <i>et al.</i> (2020)	Game (Narrow)	25 discrete or continuous goal
RLBench	James <i>et al.</i> (2020)	Robotic (Diverse)	100 discrete
Meta-World	Yu <i>et al.</i> (2020a)	Robotic (Diverse)	45 discrete; continuous (e.g., goal)
MuJoCo (Todorov <i>et al.</i> , 2012) Locomotion	e.g., Finn <i>et al.</i> (2017a) and Zintgraf <i>et al.</i> (2020)	Robotic (Narrow)	e.g., 2 discrete or continuous goal

et al., 2019; Zintgraf *et al.*, 2020). While the structure between such tasks is clear, the narrowness of the distribution poses problems. For example, in this setting, task inference is generally trivial. Often, the agent can infer the parameters that define the task based on just a few transitions. This makes it hard to evaluate if a meta-RL method can learn systematical exploration behaviors to infer more sophisticated task structures. In general, the inductive bias that can be learned from

a narrow task distribution is highly tailored to the specific training distribution, which provides little help for faster acquisition of entirely new behaviors in a broader task domain. Indeed, Zhao *et al.* (2022a) find that meta-RL methods are no better than multi-task pre-training when tested on generalization to complex visual tasks like Atari games.

Consequently, we need to design benchmarks that are diverse, in addition to having clear task structures. Such a benchmark would better reflect the complex task distribution in real-world problems, and promote the design of new methods that can generalize over these challenging tasks. For example, some recent robotic benchmarks (James *et al.*, 2020; Yu *et al.*, 2020a) introduce a wide range of manipulation tasks in simulation with not only parametric diversity (such as moving an object to different goals), but also non-parametric diversity (such as picking an object and opening a window). Game benchmarks with procedurally generated environments provide another good testbed to evaluate meta-RL on broader task distributions (Nichol *et al.*, 2018b; Cobbe *et al.*, 2020; Küttler *et al.*, 2020; Staley *et al.*, 2021; Wang *et al.*, 2021), as novel environments with both diverse and clear task structures can be easily generated by following different game rules. For example, Alchemy (Wang *et al.*, 2021) is a video game benchmark with the goal of transforming stones with potions into more valuable forms, which requires the agent to strategically experiment with different hypotheses for efficient exploration and task inference. As another example, the Adaptive Agent (Ada) evaluated on XLand 2.0 learns to experiment, use tools, and navigate in a 3D open-ended environment. In addition to these simulation benchmarks, Liu *et al.* (2022c) introduce a real-world benchmark which requires systematic exploration to discover the errors in different programs for coding feedback. Existing meta-RL methods cannot yet achieve satisfactory performance on many of these more challenging benchmarks (Alver and Precup, 2020; Yu *et al.*, 2020a; Wang *et al.*, 2021; Mandi *et al.*, 2022), which shows that generalization to a wider task distribution is still an open problem, and more attention should be paid to these more challenging benchmarks to push the limit of meta-RL algorithms.

Still, both new methods and new benchmarks are still needed in meta-RL. Novel methods, such as the use of curriculum learning (Mehta

et al., 2020; Team *et al.*, 2023) and active selection of tasks in the distribution based on task descriptions (Kaddour *et al.*, 2020), will likely be needed as well to address sufficiently broad distributions. Many of these benchmarks are insufficient as well, with more work needed to improve the task distribution itself. For example, some of benchmarks with the widest task distributions are private (Team *et al.*, 2023), or not designed to test adaptation in meta-RL (Küttler *et al.*, 2020; Cobbe *et al.*, 2020). It is also possible that some of these benchmarks simply need to be more densely populated with tasks in order for few-shot adaptation to be learnable. For example, Meta-World (Yu *et al.*, 2020a) has benchmarks with 1, 10, or 45 discrete tasks; however, meta-learning an algorithm that can reliably adapt to unseen meta-test tasks may require tens of thousands of meta-training tasks (Kirsch *et al.*, 2022b). Adapting benchmarks (Benjamins *et al.*, 2021) from the related Contextual MDP literature (Hallak *et al.*, 2015) in order to remove the task identity from the state is a path forward. Procedurally generating tasks is also one promising path for benchmarks (Cobbe *et al.*, 2020; Küttler *et al.*, 2020; Wang *et al.*, 2021; Team *et al.*, 2023). When tasks have manually designed discrete variation, it can be difficult to determine whether meta-test tasks even have support within the meta-training distribution.

Generalization to OOD tasks In few-shot meta-RL, it is commonly assumed that the meta-training and meta-test tasks are drawn from the same task distribution. However, in real-world problems, we usually do not know *a priori* all the situations the agent may face during deployment, and the RL agent will likely encounter test tasks that lie outside the meta-training task distribution.

One key challenge here is that we do not know to what extent the learned inductive bias is still helpful for solving the OOD tasks. For example, in the navigation task in Figure 3.9, the learned inductive bias is an exploration policy that traverses the edge of the semicircle to find the goal first. If we consider OOD tasks of navigation to goals on the edge of a semicircle with a larger radius, then the learned exploration policy is no longer optimal, but may still help the agent explore more efficiently than from scratch. However, if the OOD tasks are navigation

to goals on a semicircle in the opposite direction, then the learned inductive bias may be actively harmful and slow down learning.

Consequently, simply using the learned inductive bias, which is commonly adopted by existing few-shot meta-RL methods, is not sufficient for OOD generalization. Instead, the agent needs to adaptively choose how to utilize or adjust the learned inductive bias according to what kinds of OOD tasks it is solving. On one hand, we want to ensure that the agent can generalize to any OOD task given enough adaptation data, even if the learned inductive bias is misspecified. In principle, PPG methods can satisfy this requirement while black-box methods cannot, echoing our discussion on the trade-off between generalization and specialization of few-shot meta-RL methods in Section 3.1. However, in practice, if the meta-testing task distribution is sufficiently dissimilar to the meta-training task distribution, then any sort of meta-learning can be catastrophic (Xiong *et al.*, 2021). On the other hand, we want to utilize as much useful information from the learned inductive bias as possible to improve learning efficiency on OOD tasks. Although recent works investigate how to improve generalization (Lan *et al.*, 2019; Lin *et al.*, 2020b; Grewal *et al.*, 2021; Xiong *et al.*, 2021; Ajay *et al.*, 2022; He *et al.*, 2022; Imagawa *et al.*, 2022; Greenberg *et al.*, 2023; Wang *et al.*, 2023) or adaptation efficiency (Fakoor *et al.*, 2020; Mendonca *et al.*, 2020; Lee and Chung, 2021) on OOD tasks with small distribution shifts, more work remains to be done on how to adaptively handle larger distribution shifts between training and test tasks. Ideally, OOD methods make use of their inductive bias where possible, and fail gracefully (e.g., defaulting to an engineered learning algorithm), where it is not possible.

6.2 Many-shot Meta-RL: Optimization Issues and Standard Benchmarks

For many-shot meta-RL (see Section 4), outer-loop optimization poses significant problems, some of which remain open. Moreover, there is a lack of standard benchmarks to compare different many-shot meta-RL methods, an important gap to fill for future work.

Optimization issues in many-shot meta-RL In many-shot meta-RL, the inner-loop updates the policy many times, which leads to a challenging optimization problem in the outer-loop due to not smooth objective surfaces (Metz *et al.*, 2021) and high computation cost. To deal with these challenges in practice, most methods use the performance after relatively few inner-loop updates compared to the full inner-loop optimization trajectory as a surrogate objective. Updating the inner-loop after only a fraction of the lifetime leads to bias in the gradient estimation, which can be detrimental to meta-learning performance (Wu *et al.*, 2018). How to tackle this optimization issue remains an open problem. One approach is to use gradient-free optimization methods such as evolution strategies (Rechenberg, 1971) as was done by Kirsch *et al.* (2022a), but its sample complexity is much worse than gradient-based optimization in settings where those are applicable.

Truncated optimization in many-shot single-task meta-RL Even if optimizing over long lifetimes was possible in the multi-task setting, in the single-task setting we still need to update the inner-loop before learning has finished in order for it to be useful. One approach to improve truncated optimization on a single task is the bootstrapped surrogate objective (Flennerhag *et al.*, 2021), which approximates an update for a longer truncation length with a bootstrapping objective on a shorter truncation length. However, this also introduces a biased meta-gradient estimation. Another solution computes meta-gradients with different numbers of updates in the inner-loop and then computes a weighted average, similar to $\text{TD}(\lambda)$ (Bonnet *et al.*, 2021). Additionally, there are still more sources of bias in this setting, such as bias from the reuse of critics between the inner- and outer-loops (Bonnet *et al.*, 2022). More research is required to choose the optimal bias-variance trade-off for meta-gradient estimators under the single-task setting.

Non-stationary optimization in many-shot single-task meta-RL Another central challenge in many-shot single-task meta-RL is the non-stationarity of the inner-loop. In multi-task meta-RL, the inner-loop revisits the same tasks multiple times, allowing the meta-learner to fit to the stationary training task distribution. In the single-task case,

however, the agent parameters keep changing, making the meta-learning problem non-stationary. Learning in a non-stationary problem is an open area of research extending beyond online meta-RL.

Benchmarks for many-shot meta-RL Many-shot meta-RL methods are mainly evaluated on a wide range of commonly used RL tasks, such as Atari (Bellemare *et al.*, 2013), classic control (Brockman *et al.*, 2016) and continuous control (Duan *et al.*, 2016a), to show that the learned RL algorithms have good generalization. However, some papers test generalization across different domains while others evaluate within a single domain, and there are no unified criteria on how to split the training and test tasks on the chosen domain(s). In the single-task setting, a benchmark for hyperparameters tuning in RL has been proposed, but focuses on a fixed and small number of discrete meta-parameters (Shala *et al.*, 2022). In the multi-task setting, to better evaluate generalization of the learned algorithms, it could be helpful to design and adopt benchmarks that choose the meta-train and meta-test tasks based on some unified standard. In this direction, a useful benchmark may even provide multiple groups of train and test tasks in order to gradually increase the difficulty and degree of transfer required, where ideally the degree of transfer is quantifiable by some measure of similarity across the MDPs (Ammar *et al.*, 2014). Moreover, the ultimate goal of many-shot meta-RL is to design general-purpose RL algorithms that can work well on any reasonable MDP. However, the exact task structure in such a distribution over all “reasonable” MDPs is still unclear and needs clarification, as some structure must be shared across these MDPs to allow for meta-learning in the first place.

6.3 Utilizing Offline Data in Meta-RL

So far the majority of research in meta-RL focuses on the setting of using online data for both the outer-loop and the inner-loop. However, when offline data is available, utilizing it properly is an effective way to reduce the need for expensive online data collection during both meta-training (the outer-loop) and adaptation (the inner-loop). Depending on which kind of data is available for the outer-loop and inner-loop respectively,

we have four different settings, depicted in Figure 6.2. Apart from fully online meta-RL, the remaining three settings are still underexplored, and we discuss them below.

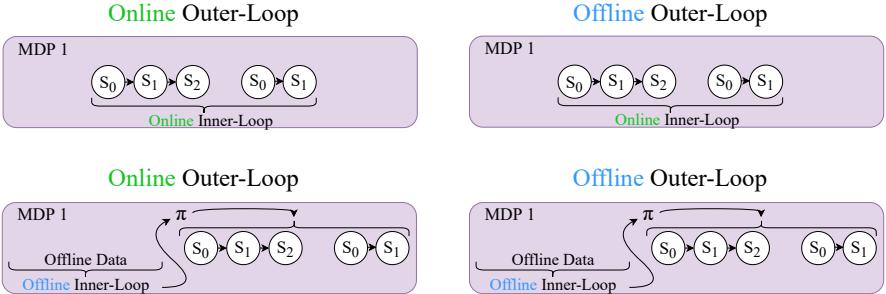


Figure 6.2: The four different settings using online or offline data. Using an online outer-loop and online inner-loop is the standard setting. Using an offline outer-loop with offline inner-loop allows for safe meta-learning and safe-adaptation by forgoing all exploration, but places additional demands on the data provided. Using an offline outer-loop with an online inner-loop allows for safe meta-learning of exploration behavior, but introduces difficulties such as extensive distribution shift. Using an online outer-loop with offline inner-loop allows for meta-learning offline RL algorithms without distribution shift, but still requires online samples during meta-learning.

Offline outer-loop and offline inner-loop Under this setting, the agent can only adapt with offline data, and learn to optimize its adaptation strategy also with offline data in the outer-loop (Mitchell *et al.*, 2020; Lee *et al.*, 2020a; Li *et al.*, 2021c; Mitchell *et al.*, 2021; Lin *et al.*, 2022; Yuan and Lu, 2022). This is particularly suitable for scenarios where online exploration and data collection is costly and even dangerous, while offline logs of historical behaviors are abundant, such as in robotics and industrial control (Levine *et al.*, 2020). However, the offline setting also introduces new challenges. First, as in standard offline RL, returns must be estimated solely from offline data, creating issues such as the overestimation of returns (Fujimoto *et al.*, 2019; Levine *et al.*, 2020). Second, and unique to meta-RL, instead of doing online exploration to collect the data, \mathcal{D} , for adaptation, we can only adapt with whatever offline data is provided for each task. Consequently, the adaptation performance critically depends on how informative the offline data is

about the task. Existing works mainly investigate this offline setting on simple task distributions where task identity can be easily inferred from a few randomly sampled transitions in the offline data. However, how to adapt with offline data on more complicated task distributions (such as those discussed in Section 6.1), and how different offline data collection schemes may influence the performance of offline adaptation, remain open questions.

Offline outer-loop and online inner-loop In this setting, the agent learns to adapt with *online* data, by meta-training on purely *offline* data (Dorfman *et al.*, 2021; Pong *et al.*, 2022; Ghosh *et al.*, 2022). Compared to the fully offline setting, this setting is more suitable for the scenarios where few-shot online adaptation is allowed during deployment. Online adaptation tackles the aforementioned problem of limited exploration when adapting with only offline data, but it also introduces a new challenge: how can we learn a systematic exploration policy from offline data collected by some unknown policies? Usually, the desired behaviors of the exploration policy are not covered in the offline data, which is generally collected in a task-specific way. This creates a distribution shift between the exploration policy we want to learn and the offline data we can learn from. This shift can be especially problematic if the offline data was collected only with an expert that has access to the ground-truth task (Rafailov *et al.*, 2021). In particular, it can introduce a problem of ambiguity in the identity of the MDP (Dorfman *et al.*, 2021). To tackle these challenges, existing works make additional assumptions on the data collection scheme, which strictly speaking, violate the offline meta-training setting and thus limit their application. For example, Pong *et al.* (2022) allow for some online meta-training, but only with unsupervised interaction, and they learn a reward function to generate supervision for this unsupervised online interaction. Additionally, Rafailov *et al.* (2021) make assumptions about the reward that enable easy reward relabeling when swapping offline trajectories between tasks. How to relax offline assumptions to make this setting more applicable remains an open problem.

Online outer-loop and offline inner-loop Under this setting, the agent can only adapt with offline data. However, online data is available in the outer-loop to help the agent learn what is a good offline adaptation strategy, which may be easier to learn than in an offline outer-loop setting. In other words, the agent is learning to do offline RL via online RL. This setting is appealing for two reasons: (1) It maintains the benefits of using solely offline data for adaptation, while being easier to meta-train than the fully offline setting. (2) It is difficult to design good offline RL algorithms (Levine *et al.*, 2020), and meta-RL provides a promising approach to automating this process. A couple of existing methods do combine an offline inner-loop with an online outer-loop (Prat and Johns, 2021; Dance *et al.*, 2021). However, these approaches both allow for additional online adaptation after the offline adaptation in the inner-loop, and the offline data either contains only observations (Dance *et al.*, 2021), or conditions on additional expert actions (Prat and Johns, 2021). One method uses permutation-invariant memory to enable an off-policy inner-loop, but only evaluates with data collected from prior policies (Imagawa *et al.*, 2022). Consequently, offline RL via online RL remains an interesting setting for future work with the potential for designing more effective offline RL algorithms by meta-learning in both the few-shot and many-shot settings.

6.4 Limitations

So far we have presented a positive case for the use and development of meta-RL. While meta-RL can confer many advantages, as a tool for solving problems, it also presents several trade-offs. At this point, we take a step back to discuss four limitations of meta-RL from a more critical perspective.

While meta-RL can enable sample-efficient learning at deployment, it does so at the expense of increased sample complexity during meta-training. For this reason, meta-RL is only applicable when upfront data collection is relatively cheap, or adaption during deployment is prohibitively expensive. For example, the trade-off presented by meta-learning makes sense when a simulator is available for meta-training, or when adaptation needs to be efficient and frequent after meta-training.

While this limitation is restrictive, it is not significantly more restrictive than the standard use case for reinforcement learning.

Meta-RL additionally presents a trade-off between transferability and interpretability on one hand and sample efficiency and engineering burden on the other. While meta-RL methods may enable adaptation that is more sample-efficient than a manually engineered alternative, the insights produced by such systems may be less interpretable and transferable to novel problems. For example, in the many-shot setting, it may be possible to meta-learn a general-purpose RL algorithm that is better than state-of-the-art algorithms, but even so, the exact mechanisms improving performance may be unclear. Consider meta-learning an objective function that is parameterized as a black-box. The meta-learned objective function may have a simple and interpretable form, but extricating this form from the weights and biases of a neural network is difficult. Meta-RL replaces engineering effort and domain expertise with computation, and in doing so, trades-off interpretability for performance.

In the few-shot setting (Section 3), it is common to adapt to tasks using gradient updates in the inner-loop. However, whether meta-learning to account for this procedure is worthwhile can depend on the particular task distribution. For example, when there is a limited number of tasks or limited amount of data in the inner-loop, it may even be preferable to have MAML perform no task adaptation during meta-training and only fine-tune at meta-test time (Gao and Sener, 2020). (In the supervised setting, it has likewise been shown that fine-tuning, only after meta-training, outperforms learning to fine-tune, when meta-training a black-box method; Beck *et al.*, 2025.) Additionally, if the task distribution never requires different actions for the same state in different tasks, e.g., because the state-space does not overlap, then adaptation may not be needed and multi-task training may be sufficient. In such a setting, even if more adaptation is useful for generalization, meta-RL may confer little to no advantage compared to fine-tuning (Mandi *et al.*, 2022). Moreover, even if adaptation to each task is needed, and sufficient data is provided, as is typically assumed, meta-RL can actually confer a disadvantage. Specifically, test-time adaptation can decrease performance on some tasks in practice (Deleu and Bengio, 2018), and training

from scratch can yield higher returns (Xiong *et al.*, 2021). (Similar observations have been made in the meta-supervised setting as well; Triantafillou *et al.*, 2020.) In fact, if the meta-testing task distribution is sufficiently dissimilar to the meta-training task distribution, then any sort of meta-learning can be catastrophic, since the agent learns incorrect inductive biases (Xiong *et al.*, 2021).

Finally, while the meta-RL literature develops many specialized methods, the problem setting may not require such methods. Consider that the meta-RL problem can be considered a particular type of POMDP, as discussed in Section 3.5. Given this, it is reasonable to question whether specialized methods are needed for meta-RL at all. In the zero-shot setting, recent work has suggested that many complex and specialized methods may be unnecessary. For example, some task-inference methods can be nearly matched by well-tuned end-to-end recurrent networks (Ni *et al.*, 2022); complicated task-inference parameterizations and supervision can be outperformed by hypernetworks trained end-to-end (Beck *et al.*, 2022); some task inference methods (Humplik *et al.*, 2019; Fu *et al.*, 2021) can be seen as applications of more general POMDP methods for inferring a hidden state (Moreno *et al.*, 2018; Guo *et al.*, 2018); and complicated task-inference reward bonuses for exploration may be just as effectively replaced by more general exploration methods for POMDPs (Yin *et al.*, 2021) applied to the meta-RL problem setting (Zintgraf *et al.*, 2021c). Even for complicated task distributions, methods designed for MDPs and more general POMDPs, such as curriculum learning, distillation, and transformer architectures, can be sufficient to enable meta-learning (Team *et al.*, 2023). Still, even if specialized methods are not strictly necessary, such methods can enable more efficient meta-learning. Moreover, whether meta-learning methods are developed explicitly or not, meta-learning will at the very least be an emergent phenomenon of any capable and general agent. For this reason, insights from meta-RL should assist practitioners in developing and reasoning about such systems.

7

Conclusion

In this survey, we presented a survey of meta-RL research focused on two major categories of algorithms as well as applications. We found the majority of research focused on the few-shot multi-task setting, where the objective is to learn an RL algorithm that adapts to new tasks from a known task distribution rapidly using as few samples as possible. We discussed the strengths and weaknesses of the few-shot algorithms, which generally fall in the categories of parameterized policy gradient, black box, and task inference methods. A central topic in using these methods is how to explore the environment to collect that data. We identified the different exploration strategies in the literature and discussed when each of them are applicable. Besides meta-RL in the few-shot setting, a rising topic in meta-RL looks at algorithms in the many-shot setting, where two distinct problems are considered: the generalization to broader task distributions and faster learning on a single task. We found the methods for these two seemingly opposite problems to be surprisingly similar, as they are often based on augmenting standard RL algorithms with learned components. We presented promising applications of meta-RL, especially those in robotics, where meta-RL is starting to enable significant sample efficiency gains in e.g.,

sim-to-real transfer. The sample efficiency of RL algorithms is a major blocker in learning controllers for real-world applications. Therefore, if meta-RL delivers on the promise of sample efficient adaptation, it would enable a wide variety of applications. In order to push meta-RL further and enable new applications, we found that broader and more diverse task distributions need to be developed for training and testing the meta-RL algorithms. With promising applications in sight and a range of open problems awaiting solutions, we expect meta-RL research to continue to actively grow.

Appendix

A

List of Venues Surveyed

This survey is primarily based on the meta-RL research presented in the following conferences and workshops for the years from 2017 to 2022:

- International Conference on Learning Representations (ICLR)
- Conference on Neural Information Processing Systems (NeurIPS)
- International Conference on Machine Learning (ICML)
- Autonomous Agents and Multiagent Systems (AAMAS)
- Annual AAAI Conference on Artificial Intelligence (AAAI)
- Conference on Robot Learning (CoRL)
- Robotics: Science and Systems (RSS)
- International Conference on Intelligent Robots and Systems (IROS)
- NeurIPS Workshop on Meta-Learning
- ICLR Workshop on Meta-Learning

References

- Agarwal, A., S. Kakade, A. Krishnamurthy, and W. Sun. (2020). “Flambe: Structural complexity and representation learning of low rank mdps”. *Advances in neural information processing systems*. 33: 20095–20107.
- Ajay, A., D. Ghosh, S. Levine, P. Agrawal, and A. Gupta. (2022). “Distributionally Adaptive Meta Reinforcement Learning”. In: *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*.
- Akkaya, I., M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.* (2019). “Solving rubik’s cube with a robot hand”. *arXiv preprint arXiv:1910.07113*.
- Akuzawa, K., Y. Iwasawa, and Y. Matsuo. (2021). “Estimating Disentangled Belief about Hidden State and Hidden Task for Meta-Reinforcement Learning”. In: *Learning for Dynamics and Control*. PMLR. 73–86.
- Alemi, A. A., I. Fischer, J. V. Dillon, and K. Murphy. (2017). “Deep Variational Information Bottleneck”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=HyxQzBceg>.
- Alet, F., M. F. Schneider, T. Lozano-Perez, and L. P. Kaelbling. (2020). “Meta-learning curiosity algorithms”. In: *International Conference on Learning Representations*.

- Almeida, D., C. Winter, J. Tang, and W. Zaremba. (2021). “A generalizable approach to learning optimizers”. *arXiv preprint arXiv:2106.00958*.
- Alver, S. and D. Precup. (2020). “A Brief Look at Generalization in Visual Meta-Reinforcement Learning”. In: *4th Lifelong Machine Learning Workshop at ICML 2020*. URL: <https://openreview.net/forum?id=WrCFtzVEwn>.
- Ammar, H. B., E. Eaton, M. E. Taylor, D. C. Mocanu, K. Driessens, G. Weiss, and K. Tuyls. (2014). “An automated measure of mdp similarity for transfer in reinforcement learning”. In: *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Andrychowicz, M., M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. (2016). “Learning to learn by gradient descent by gradient descent”. *Advances in neural information processing systems*.
- Anne, T., J. Wilkinson, and Z. Li. (2021). “Meta-Learning for Fast Adaptive Locomotion with Uncertainties in Environments and Robot Dynamics”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 4568–4575.
- Arndt, K., M. Hazara, A. Ghadirzadeh, and V. Kyrki. (2020). “Meta Reinforcement Learning for Sim-to-real Domain Adaptation”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2725–2731. DOI: [10.1109/ICRA40945.2020.9196540](https://doi.org/10.1109/ICRA40945.2020.9196540).
- Arumugam, D. and S. Singh. (2022). “Planning to the Information Horizon of BAMDPs via Epistemic State Abstraction”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=7eUOC9fEIRO>.
- Aubret, A., L. Matignon, and S. Hassas. (2019). “A survey on intrinsic motivation in reinforcement learning”. *arXiv preprint arXiv:1908.06976*.
- Baxter, J. (2000). “A model of inductive bias learning”. *Journal of artificial intelligence research*. 12: 149–198.

- Bechtle, S., A. Molchanov, Y. Chebotar, E. Grefenstette, L. Righetti, G. Sukhatme, and F. Meier. (2021). “Meta Learning via Learned Loss”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE Computer Society. 4161–4168.
- Beck, J., K. Ciosek, S. Devlin, S. Tschiatschek, C. Zhang, and K. Hofmann. (2020). “AMRL: Aggregated Memory For Reinforcement Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Bkl7bREtDr>.
- Beck, J., M. Jackson, R. Vuorio, and S. Whiteson. (2022). “Hypernetworks in Meta-Reinforcement Learning”. *CoRL*.
- Beck, J., M. Jackson, R. Vuorio, Z. Xiong, and S. Whiteson. (2024). “SplAgger: Split Aggregation for Meta-Reinforcement Learning”. *Reinforcement Learning Conference*.
- Beck, J., S. Surana, M. McAuliffe, O. Bent, T. D. Barrett, J. J. G. Luis, and P. Duckworth. (2025). “Metalic: Meta-Learning In-Context with Protein Language Models”. In: *The Thirteenth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=TUKt7ag0qq>.
- Beck, J., R. Vuorio, Z. Xiong, and S. Whiteson. (2023). “Recurrent Hypernetworks are Surprisingly Strong in Meta-RL”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=pefAAzu8an>.
- Belkhale, S., R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine. (2021). “Model-based meta-reinforcement learning for flight with suspended payloads”. *IEEE Robotics and Automation Letters*. 6(2): 1471–1478.
- Bellec, G., D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. (2018). “Long short-term memory and Learning-to-learn in networks of spiking neurons”. *NeurIPS*.
- Bellemare, M. G., S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang. (2020). “Autonomous navigation of stratospheric balloons using reinforcement learning”. *Nature*. 588(7836): 77–82.
- Bellemare, M. G., Y. Naddaf, J. Veness, and M. Bowling. (2013). “The arcade learning environment: An evaluation platform for general agents”. *Journal of Artificial Intelligence Research*. 47: 253–279.

- Benjamins, C., T. Eimer, F. Schubert, A. Biedenkapp, B. Rosenhahn, F. Hutter, and M. Lindauer. (2021). “CARL: A Benchmark for Contextual and Adaptive Reinforcement Learning”.
- Berseth, G., Z. Zhang, G. Zhang, C. Finn, and S. Levine. (2021). “Comps: Continual meta policy search”. *arXiv preprint arXiv:2112.04467*.
- Bonardi, A., S. James, and A. J. Davison. (2020). “Learning One-Shot Imitation from Humans without Humans”. *CVPR*.
- Bonnet, C., P. Caron, T. D. Barrett, I. Davies, and A. Laterre. (2021). “One Step at a Time: Pros and Cons of Multi-Step Meta-Gradient Reinforcement Learning”. *5th Workshop on Meta-Learning at NeurIPS*. URL: <https://arxiv.org/abs/2111.00206>.
- Bonnet, C., L. I. Midgley, and A. Laterre. (2022). “Debiasing Meta-Gradient Reinforcement Learning by Learning the Outer Value Function”. In: *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=HFVmkrJ4J>.
- Boutilier, C., C.-w. Hsu, B. Kveton, M. Mladenov, C. Szepesvari, and M. Zaheer. (2020). “Differentiable Meta-Learning of Bandit Policies”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc. 2122–2134. URL: <https://proceedings.neurips.cc/paper/2020/file/171ae1bbb81475eb96287dd78565b38b-Paper.pdf>.
- Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. (2016). “Openai gym”. *arXiv preprint arXiv:1606.01540*.
- Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. (2020). “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*.

- Burda, Y., H. Edwards, A. Storkey, and O. Klimov. (2019). “Exploration by random network distillation”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=H1JJnR5Ym>.
- Burega, B., J. D. Martin, and M. Bowling. (2022). “Learning to Prioritize Planning Updates in Model-based Reinforcement Learning”. In: *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=uR7ePjeB6z>.
- Chalvidal, M., T. Serre, and R. VanRullen. (2022). “Meta-Reinforcement Learning with Self-Modifying Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=cYeYzaP-5AF>.
- Chan, S. C. Y., I. Dasgupta, J. Kim, D. Kumaran, A. K. Lampinen, and F. Hill. (2022). “Transformers generalize differently from information stored in context vs in weights”. In: *Workshop on Memory in Artificial and Real Intelligence at NeurIPS*.
- Charakorn, R., P. Manoonpong, and N. Dilokthanakul. (2021). “Learning to Cooperate with Unseen Agents Through Meta-Reinforcement Learning”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 1478–1479.
- Chen, C., H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li. (2020). “Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 04. 3414–3421.
- Chen, X., J. Hu, C. Jin, L. Li, and L. Wang. (2022). “Understanding Domain Randomization for Sim-to-real Transfer”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=T8vZHIRTTrY>.
- Chen, Y., M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. Freitas. (2017). “Learning to learn without gradient descent by gradient descent”. In: *International Conference on Machine Learning*. PMLR. 748–756.

- Cheng, Y., S. Feng, J. Yang, H. Zhang, and Y. Liang. (2022). “Provable benefit of multitask representation learning in reinforcement learning”. *Advances in Neural Information Processing Systems*. 35: 31741–31754.
- Choshen, E. and A. Tamar. (2023). “ContraBAR: Contrastive Bayesian Adaptive Deep RL”. *ICML*.
- Chowdhery, A., S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. (2022). “PaLM: Scaling Language Modeling with Pathways”.
- Clavera, I., J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. (2018). “Model-Based Reinforcement Learning via Meta-Policy Optimization”. *CoRL*.
- Cobbe, K., C. Hesse, J. Hilton, and J. Schulman. (2020). “Leveraging procedural generation to benchmark reinforcement learning”. In: *International conference on machine learning*. PMLR. 2048–2056.
- Collins, L., A. Mokhtari, S. Oh, and S. Shakkottai. (2022). “Maml and anil provably learn representations”. In: *International Conference on Machine Learning*. PMLR. 4238–4310.
- Cong, L., M. Görner, P. Ruppel, H. Liang, N. Hendrich, and J. Zhang. (2020). “Self-Adapting Recurrent Models for Object Pushing from Learning in Simulation”. *IROS*.

- Dance, C. R., J. Perez, and T. Cachet. (2021). “Demonstration-Conditioned Reinforcement Learning for Few-Shot Imitation”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. *Proceedings of Machine Learning Research*. PMLR. 2376–2387. URL: <https://proceedings.mlr.press/v139/dance21a.html>.
- Dasari, S. and A. Gupta. (2020). “Transformers for One-Shot Visual Imitation”. *CoRL*. abs/2011.05970. URL: <https://arxiv.org/abs/2011.05970>.
- Deleu, T. and Y. Bengio. (2018). “The effects of negative adaptation in model-agnostic meta-learning”. *arXiv preprint arXiv:1812.02159*.
- Denevi, G., C. Ciliberto, R. Grazzi, and M. Pontil. (2019). “Learning-to-learn stochastic gradient descent with biased regularization”. In: *International Conference on Machine Learning*. PMLR. 1566–1575.
- Dennis, M., N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. (2020). “Emergent complexity and zero-shot transfer via unsupervised environment design”. *Advances in neural information processing systems*. 33: 13049–13061.
- Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.
- Dorfman, R., I. Shenfeld, and A. Tamar. (2021). “Offline Meta Reinforcement Learning–Identifiability Challenges and Effective Data Collection Strategies”. *Advances in Neural Information Processing Systems*. 34: 4607–4618.
- Duan, Y., M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. (2017). “One-Shot Imitation Learning”. In: *NIPS*.
- Duan, Y., X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. (2016a). “Benchmarking deep reinforcement learning for continuous control”. In: *International conference on machine learning*. PMLR. 1329–1338.

- Duan, Y., J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. (2016b). “RL²: Fast reinforcement learning via slow reinforcement learning”. *arXiv preprint arXiv:1611.02779*.
- Duff, M. O. and A. Barto. (2002). “Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes”. *PhD thesis*.
- Elawady, A., G. Chhablani, R. Ramrakhya, K. Yadav, D. Batra, Z. Kira, and A. Szot. (2024). “ReLIC: A Recipe for 64k Steps of In-Context Reinforcement Learning for Embodied AI”. *arXiv*.
- Ellis, B., S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. N. Foerster, and S. Whiteson. (2022). “SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning”. *arXiv preprint arXiv:2212.07489*.
- Emukpere, D., X. Alameda-Pineda, and C. Reinke. (2021). “Successor Feature Neural Episodic Control”. *arXiv preprint arXiv:2111.03110*.
- Espeholt, L., H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, *et al.* (2018). “Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures”. In: *International Conference on Machine Learning*. PMLR. 1407–1416.
- Eysenbach, B., A. Gupta, J. Ibarz, and S. Levine. (2019). “Diversity is All You Need: Learning Skills without a Reward Function”. *ICLR*.
- Fakoor, R., P. Chaudhari, S. Soatto, and A. J. Smola. (2020). “Meta-Q-Learning”. In: *International Conference on Learning Representations*.
- Fallah, A., K. Georgiev, A. Mokhtari, and A. Ozdaglar. (2020). “Provably convergent policy gradient methods for model-agnostic meta-reinforcement learning”. *arXiv preprint arXiv:2002.05135*.
- Fallah, A., K. Georgiev, A. Mokhtari, and A. Ozdaglar. (2021). “On the convergence theory of debiased model-agnostic meta-reinforcement learning”. *Advances in Neural Information Processing Systems*. 34: 3096–3107.
- Farquhar, G., S. Whiteson, and J. Foerster. (2019). “Loaded DiCE: Trading off bias and variance in any-order score function gradient estimators for reinforcement learning”. *Advances in Neural Information Processing Systems*.

- Feng, X., O. Slumbers, Z. Wan, B. Liu, S. McAleer, Y. Wen, J. Wang, and Y. Yang. (2021). “Neural auto-curricula in two-player zero-sum games”. *Advances in Neural Information Processing Systems*. 34.
- Finn, C., P. Abbeel, and S. Levine. (2017a). “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International Conference on Machine Learning*. PMLR. 1126–1135.
- Finn, C. and S. Levine. (2018). “Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm”. *ICLR*.
- Finn, C., A. Rajeswaran, S. Kakade, and S. Levine. (2019). “Online meta-learning”. In: *International Conference on Machine Learning*. PMLR. 1920–1930.
- Finn, C., T. Yu, T. Zhang, P. Abbeel, and S. Levine. (2017b). “One-Shot Visual Imitation Learning via Meta-Learning”. *CoRL*.
- Flennerhag, S., A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell. (2020). “Meta-Learning with Warped Gradient Descent”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=rkeiQlBFPB>.
- Flennerhag, S., Y. Schroecker, T. Zahavy, H. van Hasselt, D. Silver, and S. Singh. (2021). “Bootstrapped Meta-Learning”. *arXiv preprint arXiv:2109.04504*.
- Foerster, J., G. Farquhar, M. Al-Shedivat, T. Rocktäschel, E. Xing, and S. Whiteson. (2018a). “Dice: The infinitely differentiable monte carlo estimator”. In: *International Conference on Machine Learning*. PMLR. 1529–1538.
- Foerster, J. N., R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch. (2018b). “Learning with Opponent-Learning Awareness”. *AAMAS*.
- Fortunato, M., M. Tan, R. Faulkner, S. Hansen, A. P. Badia, G. Buttaltimore, C. Deck, J. Z. Leibo, and C. Blundell. (2019). “Generalization of Reinforcement Learners with Working and Episodic Memory”. *NeurIPS*.
- Fu, H., H. Tang, J. Hao, C. Chen, X. Feng, D. Li, and W. Liu. (2021). “Towards effective context for meta-reinforcement learning: an approach based on contrastive learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. No. 8. 7457–7465.

- Fujimoto, S., D. Meger, and D. Precup. (2019). “Off-Policy Deep Reinforcement Learning without Exploration”. *ICML*.
- Galashov, A., J. Schwarz, H. Kim, M. Garnelo, D. Saxton, P. Kohli, S. M. A. Eslami, and Y. W. Teh. (2019). “Meta-Learning surrogate models for sequential decision making”.
- Gao, C., Y. Jiang, and F. Chen. (2022). “Transferring Hierarchical Structures with Dual Meta Imitation Learning”. In: *6th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=MUcBYHjzqp7>.
- Gao, K. and O. Sener. (2020). “Modeling and Optimization Trade-off in Meta-learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc. 11154–11165. URL: <https://proceedings.neurips.cc/paper/2020/file/7fc63ff01769c4fa7d9279e97e307829-Paper.pdf>.
- Gao, Y., E. Sibirtseva, G. Castellano, and D. Krasic. (2019). “Fast adaptation with meta-reinforcement learning for trust modelling in human-robot interaction”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 305–312.
- Garau-Luis, J. J., Y. Miao, J. D. Co-Reyes, A. Parisi, J. Tan, E. Real, and A. Faust. (2022). “Multi-objective evolution for generalizable policy gradient algorithms”. In: *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*.
- Garcia, F. M. and P. S. Thomas. (2019). “A Meta-MDP Approach to Exploration for Lifelong Reinforcement Learning”. *NeurIPS*.
- Garnelo, M., J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. (2018). “Neural Processes”. *ICML*.
- Gerstgrasser, M. and D. C. Parkes. (2022). “Meta-RL for Multi-Agent RL: Learning to Adapt to Evolving Agents”. In: *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=0toY1f8-Iq9>.

- Ghadirzadeh, A., X. Chen, P. Poklukar, C. Finn, M. Björkman, and D. Kragic. (2021). “Bayesian Meta-Learning for Few-Shot Policy Adaptation Across Robotic Platforms”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1274–1280. DOI: [10.1109/IROS51168.2021.9636628](https://doi.org/10.1109/IROS51168.2021.9636628).
- Ghavamzadeh, M., S. Mannor, J. Pineau, A. Tamar, et al. (2015). “Bayesian reinforcement learning: A survey”. *Foundations and Trends® in Machine Learning*. 8(5-6): 359–483.
- Ghosh, D., A. Ajay, P. Agrawal, and S. Levine. (2022). “Offline rl policies should be trained to be adaptive”. In: *International Conference on Machine Learning*. PMLR. 7513–7530.
- Goldie, A. D., C. Lu, M. T. Jackson, S. Whiteson, and J. N. Foerster. (2024). “Can Learned Optimization Make Reinforcement Learning Less Difficult?” *Advances in Neural Information Processing Systems*.
- Goo, W. and S. Niekum. (2019). “One-shot learning of multi-step tasks from observation via activity localization in auxiliary video”. In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 7755–7761.
- Graves, A., G. Wayne, and I. Danihelka. (2014). “Neural Turing Machines”. *arXiv*. abs/1410.5401. URL: <http://arxiv.org/abs/1410.5401>.
- Greenberg, I., S. Mannor, G. Chechik, and E. Meirom. (2023). “Train Hard, Fight Easy: Robust Meta Reinforcement Learning”. *arXiv preprint arXiv:2301.11147*.
- Grewal, Y. S., F. de Nijs, and S. Goodwin. (2021). “Variance-Seeking Meta-Exploration to Handle Out-of-Distribution Tasks”. In: *Deep RL Workshop NeurIPS 2021*.
- Grigsby, J., L. Fan, and Y. Zhu. (2024). “AMAGO: Scalable In-Context Reinforcement Learning for Adaptive Agents”. *ICLR*.
- Grover, A., M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards. (2018). “Learning Policy Representations in Multiagent Systems”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. *Proceedings of Machine Learning Research*. PMLR. 1802–1811. URL: <https://proceedings.mlr.press/v80/grover18a.html>.

- Guez, A., D. Silver, and P. Dayan. (2013). “Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search”. *Journal of Artificial Intelligence Research*. 48: 841–883.
- Guo, Y., Q. Wu, and H. Lee. (2022). “Learning Action Translator for Meta Reinforcement Learning on Sparse-Reward Tasks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 6. 6792–6800.
- Guo, Z. D., M. G. Azar, B. Piot, B. A. Pires, and R. Munos. (2018). “Neural predictive belief representations”. *arXiv preprint arXiv:1811.06407*.
- Gupta, A., M. Lanctot, and A. Lazaridou. (2021). “Dynamic population-based meta-learning for multi-agent communication with natural language”. *NeurIPS*.
- Gupta, A., B. Eysenbach, C. Finn, and S. Levine. (2018a). “Unsupervised Meta-Learning for Reinforcement Learning”. *arXiv*. abs/1806.04640.
- Gupta, A., R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. (2018b). “Meta-Reinforcement Learning of Structured Exploration Strategies”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2018/file/4de754248c196c85ee4fbdcee89179bd-Paper.pdf>.
- Gurumurthy, S., S. Kumar, and K. Sycara. (2020). “MAME : Model-Agnostic Meta-Exploration”. In: *Proceedings of the Conference on Robot Learning*. Ed. by L. P. Kaelbling, D. Kragic, and K. Sugiura. Vol. 100. *Proceedings of Machine Learning Research*. PMLR. 910–922. URL: <https://proceedings.mlr.press/v100/gurumurthy20a.html>.
- Ha, D., A. Dai, and Q. V. Le. (2017). “Hypernetworks”. In: *International Conference on Learning Representation (ICLR)*.
- Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine. (2018). “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 1861–1870.
- Hallak, A., D. D. Castro, and S. Mannor. (2015). “Contextual Markov Decision Processes”.

- Harrison, J., A. Sharma, R. Calandra, and M. Pavone. (2018). “Control adaptation via meta-learning dynamics”. In: *Workshop on Meta-Learning at NeurIPS*. Vol. 2018.
- Hausknecht, M. J. and P. Stone. (2015). “Deep Recurrent Q-Learning for Partially Observable MDPs”. In: *AAAI Fall Symposia*.
- He, J. Z.-Y., Z. Erickson, D. S. Brown, A. Raghunathan, and A. Dragan. (2022). “Learning Representations that Enable Generalization in Assistive Tasks”. In: *6th Annual Conference on Robot Learning*. URL: https://openreview.net/forum?id=b88HF4vd_ej.
- Hebb, D. O. (1949). *The organization of behavior; a neuropsychological theory*.
- Heess, N., J. J. Hunt, T. P. Lillicrap, and D. Silver. (2015). “Memory-based control with recurrent neural networks”. *NIPS Deep Reinforcement Learning Workshop*.
- Hejna III, D. J. and D. Sadigh. (2022). “Few-Shot Preference Learning for Human-in-the-Loop RL”. In: *6th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=IKC5TfXLuW0>.
- Hessel, M., H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. Van Hasselt. (2019). “Multi-task deep reinforcement learning with popart”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. No. 01. 3796–3803.
- Hiraoka, T., T. Imagawa, V. Tangkaratt, T. Osa, T. Onishi, and Y. Tsuruoka. (2021). “Meta-model-based meta-policy optimization”. In: *Asian Conference on Machine Learning*. PMLR. 129–144.
- Hochreiter, S., A. S. Younger, and P. R. Conwell. (2001). “Learning to learn using gradient descent”. In: *International conference on artificial neural networks*. Springer. 87–94.
- Hospedales, T., A. Antoniou, P. Micaelli, and A. Storkey. (2020). “Meta-learning in neural networks: A survey”. *arXiv preprint arXiv:2004.05439*.
- Houthooft, R., Y. Chen, P. Isola, B. Stadie, F. Wolski, O. Jonathan Ho, and P. Abbeel. (2018). “Evolved Policy Gradients”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2018/file/7876acb66640bad41f1e1371ef30c180-Paper.pdf>.

- Hu, J., X. Chen, C. Jin, L. Li, and L. Wang. (2021). “Near-optimal representation learning for linear bandits and linear rl”. In: *International Conference on Machine Learning*. PMLR. 4349–4358.
- Huisman, M., J. N. Van Rijn, and A. Plaat. (2021). “A survey of deep meta-learning”. *Artificial Intelligence Review*. 54(6): 4483–4541.
- Humplik, J., A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess. (2019). “Meta reinforcement learning as task inference”. *arXiv preprint arXiv:1905.06424*.
- Imagawa, T., T. Hiraoka, and Y. Tsuruoka. (2022). “Off-Policy Meta-Reinforcement Learning With Belief-Based Task Inference”. *IEEE Access*. 10: 49494–49507.
- Jabri, A., K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn. (2019). “Unsupervised curricula for visual meta-reinforcement learning”. *Advances in Neural Information Processing Systems*. 32.
- Jackson, M. T., M. Jiang, J. Parker-Holder, R. Vuorio, C. Lu, G. Farquhar, S. Whiteson, and J. N. Foerster. (2023). “Discovering General Reinforcement Learning Algorithms with Adversarial Environment Design”. *arXiv preprint arXiv:2310.02782*.
- Jaderberg, M., V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. (2016). “Reinforcement learning with unsupervised auxiliary tasks”. *arXiv preprint arXiv:1611.05397*.
- James, S., M. Bloesch, and A. J. Davison. (2018). “Task-Embedded Control Networks for Few-Shot Imitation Learning”. In: *CoRL*.
- James, S., Z. Ma, D. R. Arrojo, and A. J. Davison. (2020). “Rlbench: The robot learning benchmark & learning environment”. *IEEE Robotics and Automation Letters*. 5(2): 3019–3026.
- Jang, E., A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. (2021). “BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning”. In: *5th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=8kbp23tSGYv>.
- Jin, C., Z. Yang, Z. Wang, and M. I. Jordan. (2020). “Provably efficient reinforcement learning with linear function approximation”. In: *Conference on Learning Theory*. PMLR. 2137–2143.

- Kaddour, J., S. Saemundsson, and M. Deisenroth (he/him). (2020). “Probabilistic Active Meta-Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc. 20813–20822. URL: <https://proceedings.neurips.cc/paper/2020/file/ef0d17b3bdb4ee2aa741ba28c7255c53-Paper.pdf>.
- Kamienny, P.-A., M. Pirotta, A. Lazaric, T. Lavril, N. Usunier, and L. Denoyer. (2020). “Learning adaptive exploration strategies in dynamic environments through informed policy regularization”. *arXiv preprint arXiv:2005.02934*.
- Katharopoulos, A., A. Vyas, N. Pappas, and F. Fleuret. (2020). “Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention”. *ICML*.
- Kaushik, R., T. Anne, and J.-B. Mouret. (2020). “Fast online adaptation in robotics through meta-learning embeddings of simulated priors”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 5269–5276.
- Khodak, M., M.-F. F. Balcan, and A. S. Talwalkar. (2019). “Adaptive gradient-based meta-learning methods”. *Advances in Neural Information Processing Systems*. 32.
- Kim, D. K., M. Liu, M. D. Riemer, C. Sun, M. Abdulhai, G. Habibi, S. Lopez-Cot, G. Tesauro, and J. How. (2021). “A policy gradient algorithm for learning to learn in multiagent reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 5541–5550.
- Kirk, R., A. Zhang, E. Grefenstette, and T. Rocktäschel. (2023). “A survey of zero-shot generalisation in deep reinforcement learning”. *Journal of Artificial Intelligence Research*. 76: 201–264.
- Kirsch, L., S. Flennerhag, H. van Hasselt, A. Friesen, J. Oh, and Y. Chen. (2022a). “Introducing Symmetries to Black Box Meta Reinforcement Learning”. *AAAI*.
- Kirsch, L., J. Harrison, C. D. Freeman, J. Sohl-Dickstein, and J. Schmidhuber. (2023). “Towards General-Purpose In-Context Learning Agents”. In: *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*.

- Kirsch, L., J. Harrison, J. Sohl-Dickstein, and L. Metz. (2022b). “General-purpose in-context learning by meta-learning transformers”. *arXiv*.
- Kirsch, L., S. van Steenkiste, and J. Schmidhuber. (2019). “Improving generalization in meta reinforcement learning using learned objectives”. *arXiv preprint arXiv:1910.04098*.
- Korshunova, I., J. Degrave, J. Dambre, A. Gretton, and F. Huszar. (2020). “Exchangeable Models in Meta Reinforcement Learning”. In: *4th Lifelong Machine Learning Workshop at ICML 2020*. URL: <https://openreview.net/forum?id=TZFlzejFPT>.
- Kumar, A., Z. Fu, D. Pathak, and J. Malik. (2021a). “RMA: Rapid Motor Adaptation for Legged Robots”. *RSS*. abs/2107.04034.
- Kumar, S., I. Dasgupta, J. D. Cohen, N. D. Daw, and T. L. Griffiths. (2020). “Meta-Learning of Compositional Task Distributions in Humans and Machines”. In: *4th Workshop on Meta-Learning at NeurIPS*.
- Kumar, V. C. V., S. Ha, and C. K. Liu. (2021b). “Error-Aware Policy Learning: Zero-Shot Generalization in Partially Observable Dynamic Environments”. In: *Proceedings of Robotics: Science and Systems*. Virtual. DOI: [10.15607/RSS.2021.XVII.065](https://doi.org/10.15607/RSS.2021.XVII.065).
- Küttler, H., N. Nardelli, A. Miller, R. Raileanu, M. Selvatici, E. Grefenstette, and T. Rocktäschel. (2020). “The nethack learning environment”. *Advances in Neural Information Processing Systems*. 33: 7671–7684.
- Kveton, B., M. Konobeev, M. Zaheer, C.-W. Hsu, M. Mladenov, C. Boutilier, and C. Szepesvari. (2021). “Meta-Thompson Sampling”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. *Proceedings of Machine Learning Research*. PMLR. 5884–5893. URL: <https://proceedings.mlr.press/v139/kveton21a.html>.
- Lan, L., Z. Li, X. Guan, and P. Wang. (2019). “Meta reinforcement learning with task embedding and shared policy”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2794–2800.
- Lan, Q., A. R. Mahmood, S. Yan, and Z. Xu. (2023). “Learning to Optimize for Reinforcement Learning”. *arXiv preprint arXiv:2302.01470*.

- Lange, R. T. and H. Sprekeler. (2020). “Learning not to learn: Nature versus nurture in silico”. In: *4th Workshop on Meta-Learning at NeurIPS*.
- Laskin, M., L. Wang, J. Oh, E. Parisotto, S. Spencer, R. Steigerwald, D. Strouse, S. Hansen, A. Filos, E. Brooks, *et al.* (2023). “In-context reinforcement learning with algorithm distillation”. *ICLR*.
- Lee, B., J. Lee, P. Vrancx, D. Kim, and K.-E. Kim. (2020a). “Batch reinforcement learning with hyperparameter gradients”. In: *International Conference on Machine Learning*. PMLR. 5725–5735.
- Lee, G., B. Hou, A. Mandalika, J. Lee, S. Choudhury, and S. S. Srinivasa. (2019). “Bayesian Policy Optimization for Model Uncertainty”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=SJGvns0qK7>.
- Lee, J., A. Xie, A. Pacchiano, Y. Chandak, C. Finn, O. Nachum, and E. Brunskill. (2024). “Supervised pretraining can learn in-context reinforcement learning”. *Advances in Neural Information Processing Systems*. 36.
- Lee, K., Y. Seo, S. Lee, H. Lee, and J. Shin. (2020b). “Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning”. *ICML*.
- Lee, S. and S.-Y. Chung. (2021). “Improving Generalization in Meta-RL with Imaginary Tasks from Latent Dynamics Mixture”. *Advances in Neural Information Processing Systems*. 34.
- Levine, S., A. Kumar, G. Tucker, and J. Fu. (2020). “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. *arXiv preprint arXiv:2005.01643*.
- Li, J., T. Lu, X. Cao, Y. Cai, and S. Wang. (2021a). “Meta-imitation learning by watching video demonstrations”. In: *International Conference on Learning Representations*.
- Li, K. and J. Malik. (2016). “Learning to optimize”. *arXiv preprint arXiv:1606.01885*.
- Li, K. and J. Malik. (2017). “Learning to Optimize”. *ICLR*.

- Li, K., A. Gupta, A. Reddy, V. H. Pong, A. Zhou, J. Yu, and S. Levine. (2021b). “MURAL: Meta-Learning Uncertainty-Aware Rewards for Outcome-Driven Reinforcement Learning”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. *Proceedings of Machine Learning Research*. PMLR. 6346–6356. URL: <https://proceedings.mlr.press/v139/li21g.html>.
- Li, L., R. Yang, and D. Luo. (2021c). “FOCAL: Efficient Fully-Offline Meta-Reinforcement Learning via Distance Metric Learning and Behavior Regularization”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: <https://openreview.net/forum?id=8cpHIfgY4Dj>.
- Li, Z., F. Zhou, F. Chen, and H. Li. (2017). “Meta-SGD: Learning to Learn Quickly for Few Shot Learning”. *CoRR*. abs/1707.09835. URL: <http://arxiv.org/abs/1707.09835>.
- Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. (2016). “Continuous control with deep reinforcement learning.” In: *ICLR (Poster)*. URL: <http://arxiv.org/abs/1509.02971>.
- Lin, S., J. Wan, T. Xu, Y. Liang, and J. Zhang. (2022). “Model-Based Offline Meta-Reinforcement Learning with Regularization”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=EBn0uInJZWh>.
- Lin, X., Harjatin, S. Baweja, G. Kantor, and D. Held. (2019). “Adaptive Auxiliary Task Weighting for Reinforcement Learning”. In: *NeurIPS*.
- Lin, Z., G. Thomas, G. Yang, and T. Ma. (2020a). “Model-based Adversarial Meta-Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*.
- Lin, Z., G. Thomas, G. Yang, and T. Ma. (2020b). “Model-based adversarial meta-reinforcement learning”. *Advances in Neural Information Processing Systems*. 33: 10161–10173.
- Liotet, P., F. Vidaich, A. M. Metelli, and M. Restelli. (2022). “Lifelong hyper-policy optimization with multiple importance sampling regularization”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 7. 7525–7533.

- Liu, B., X. Feng, J. Ren, L. Mai, R. Zhu, H. Zhang, J. Wang, and Y. Yang. (2022a). “A Theoretical Understanding of Gradient Bias in Meta-Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=p9zeOtKQXKs>.
- Liu, B., X. Feng, J. Ren, L. Mai, R. Zhu, H. Zhang, J. Wang, and Y. Yang. (2022b). “A theoretical understanding of gradient bias in meta-reinforcement learning”. *Advances in Neural Information Processing Systems*. 35: 31059–31072.
- Liu, E. Z., A. Raghunathan, P. Liang, and C. Finn. (2021). “Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices”. In: *International Conference on Machine Learning*. PMLR. 6925–6935.
- Liu, E. Z., M. P. Stephan, A. Nie, C. J. Piech, E. Brunskill, and C. Finn. (2022c). “Giving Feedback on Interactive Student Programs with Meta-Exploration”. In: *Advances in Neural Information Processing Systems*.
- Liu, H., R. Socher, and C. Xiong. (2019). “Taming maml: Efficient unbiased meta-reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 4061–4071.
- Lou, B., N. Zhao, and J. Wang. (2021). “Meta-learning from sparse recovery”. In: *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*.
- Lu, C., J. Kuba, A. Letcher, L. Metz, C. Schroeder de Witt, and J. Foerster. (2022a). “Discovered Policy Optimisation”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc. 16455–16468. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/688c7a82e31653e7c256c6c29fd3b438-Paper-Conference.pdf.
- Lu, C., T. Willi, A. Letcher, and J. N. Foerster. (2022b). “Adversarial Cheap Talk”. In: *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=VE6FEZc0ppH>.

- Lu, C., T. Willi, C. A. S. De Witt, and J. Foerster. (2022c). “Model-free opponent shaping”. In: *International Conference on Machine Learning*. PMLR. 14398–14411.
- Luketina, J., S. Flennerhag, Y. Schroecker, D. Abel, T. Zahavy, and S. Singh. (2022). “Meta-Gradients in Non-Stationary Environments”. In: *ICLR Workshop on Agent Learning in Open-Endedness*. URL: <https://openreview.net/forum?id=SlzBXwZIZ9>.
- Luna Gutierrez, R. and M. Leonetti. (2020). “Information-theoretic task selection for meta-reinforcement learning”. *Advances in Neural Information Processing Systems*. 33: 20532–20542.
- Luo, F.-M., S. Jiang, Y. Yu, Z. Zhang, and Y.-F. Zhang. (2022). “Adapt to environment sudden changes by learning a context sensitive policy”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 7. 7637–7646.
- Mandi, Z., P. Abbeel, and S. James. (2022). “On the Effectiveness of Fine-tuning Versus Meta-RL for Robot Manipulation”. In: *CoRL 2022 Workshop on Pre-training Robot Learning*.
- Mao, J., J. Foerster, T. Rocktäschel, M. Al-Shedivat, G. Farquhar, and S. Whiteson. (2019). “A baseline for any order gradient estimation in stochastic computation graphs”. In: *International Conference on Machine Learning*. PMLR. 4343–4351.
- Mehta, B., T. Deleu, S. C. Raparthy, C. J. Pal, and L. Paull. (2020). “Curriculum in Gradient-Based Meta-Reinforcement Learning”. In: *Beyond Tabula Rasa in RL Workshop at ICLR*.
- Meier, R. and A. Mujika. (2022). “Open-Ended Reinforcement Learning with Neural Reward Functions”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: https://openreview.net/forum?id=NL05_JGVg99.
- Melo, L. C. (2022). “Transformers are Meta-Reinforcement Learners”.
- Mendonca, R., X. Geng, C. Finn, and S. Levine. (2020). “Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling”. *arXiv preprint arXiv:2006.07178*.
- Mendonca, R., A. Gupta, R. Kralev, P. Abbeel, S. Levine, and C. Finn. (2019). “Guided Meta-Policy Search”. *NeurIPS*.
- Metz, L., C. D. Freeman, S. S. Schoenholz, and T. Kachman. (2021). “Gradients are not all you need”. *arXiv preprint arXiv:2111.05803*.

- Metz, L., N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-Dickstein. (2019). “Understanding and correcting pathologies in the training of learned optimizers”. In: *International Conference on Machine Learning*. PMLR. 4556–4565.
- Miconi, T., A. Rawal, J. Clune, and K. O. Stanley. (2019). “Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=r1lrAiA5Ym>.
- Miconi, T., K. Stanley, and J. Clune. (2018). “Differentiable plasticity: training plastic neural networks with backpropagation”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. *Proceedings of Machine Learning Research*. PMLR. 3559–3568. URL: <https://proceedings.mlr.press/v80/miconi18a.html>.
- Miki, T., J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. (2022). “Learning robust perceptive locomotion for quadrupedal robots in the wild”. *Science Robotics*. 7(62): eabk2822.
- Mishra, N., M. Rohaninejad, X. Chen, and P. Abbeel. (2018). “A Simple Neural Attentive Meta-Learner”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=B1DmUzWAW>.
- Mitchell, E., R. Rafailov, X. B. Peng, S. Levine, and C. Finn. (2020). “Offline Meta-Reinforcement Learning with Advantage Weighting”. *CoRR*. abs/2008.06043. URL: <https://arxiv.org/abs/2008.06043>.
- Mitchell, E., R. Rafailov, X. B. Peng, S. Levine, and C. Finn. (2021). “Offline Meta-Reinforcement Learning with Advantage Weighting”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. *Proceedings of Machine Learning Research*. PMLR. 7780–7791. URL: <https://proceedings.mlr.press/v139/mitchell21a.html>.
- Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. (2016). “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. PMLR. 1928–1937.

- Moreno, P., J. Humplík, G. Papamakarios, B. A. Pires, L. Buesing, N. Heess, and T. Weber. (2018). “Neural belief states for partially observed domains”. In: *NeurIPS 2018 workshop on reinforcement learning under partial observability*.
- Mu, Y., Y. Zhuang, F. Ni, B. Wang, J. Chen, J. Hao, and P. Luo. (2022). “DOMINO: Decomposed Mutual Information Optimization for Generalized Context in Meta-Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: https://openreview.net/forum?id=CJGUABT_COm.
- Mutti, M., M. Mancassola, and M. Restelli. (2022). “Unsupervised reinforcement learning in multiple environments”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 7. 7850–7858.
- Nagabandi, A., I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. (2019a). “Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning”. In: *International Conference on Learning Representations*.
- Nagabandi, A., I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. (2019b). “Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=HyztsoC5Y7>.
- Nagabandi, A., C. Finn, and S. Levine. (2019c). “Deep Online Learning Via Meta-Learning: Continual Adaptation for Model-Based RL”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=HyxAfnA5tm>.
- Najarro, E. and S. Risi. (2020). “Meta-Learning through Hebbian Plasticity in Random Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc. 20719–20731. URL: <https://proceedings.neurips.cc/paper/2020/file/ee23e7ad9b473ad072d57aaa9b2a5222-Paper.pdf>.

- Nguyen, H. H., A. Baisero, D. Wang, C. Amato, and R. Platt. (2022). “Leveraging Fully Observable Policies for Learning under Partial Observability”. In: *6th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=pn-HOPBioUE>.
- Ni, T., B. Eysenbach, and R. Salakhutdinov. (2022). “Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs”. In: *Proceedings of the 39th International Conference on Machine Learning*.
- Nichol, A., J. Achiam, and J. Schulman. (2018a). “On first-order meta-learning algorithms”. *arXiv preprint arXiv:1803.02999*.
- Nichol, A., V. Pfau, C. Hesse, O. Klimov, and J. Schulman. (2018b). “Gotta learn fast: A new benchmark for generalization in rl”. *arXiv preprint arXiv:1804.03720*.
- Oh, J., V. Chockalingam, S. P. Singh, and H. Lee. (2016). “Control of Memory, Active Perception, and Action in Minecraft”. *ICML*.
- Oh, J., M. Hessel, W. M. Czarnecki, Z. Xu, H. van Hasselt, S. Singh, and D. Silver. (2020). “Discovering reinforcement learning algorithms”. *arXiv preprint arXiv:2007.08794*.
- Packer, C., P. Abbeel, and J. E. Gonzalez. (2021). “Hindsight Task Relabelling: Experience Replay for Sparse Reward Meta-RL”. In: *Advances in Neural Information Processing Systems*. URL: <https://proceedings.neurips.cc/paper/2021/file/1454ca2270599546dfcd2a3700e4d2f1-Paper.pdf>.
- Paine, T. L., S. G. Colmenarejo, Z. Wang, S. Reed, Y. Aytar, T. Pfaff, M. W. Hoffman, G. Barth-Maron, S. Cabi, D. Budden, et al. (2018). “One-shot high-fidelity imitation: Training large-scale deep nets with rl”. *arXiv preprint arXiv:1810.05017*.
- Papoudakis, G. and S. V. Albrecht. (2020). “Variational Autoencoders for Opponent Modeling in Multi-Agent Systems”. In: *AAAI 2020 Workshop on Reinforcement Learning in Games*.
- Parisotto, E., H. F. Song, J. W. Rae, R. Pascanu, Ç. Gülçehre, S. M. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, M. M. Botvinick, N. Heess, and R. Hadsell. (2020). “Stabilizing Transformers for Reinforcement Learning”. *ICML*.
- Park, E. and J. B. Oliva. (2019). “Meta-Curvature”. *NeurIPS*.

- Park, S., O. Rybkin, and S. Levine. (2024). “METRA: Scalable Unsupervised RL with Metric-Aware Abstraction”. In: *The Twelfth International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=c5pwL0Soay>.
- Parker-Holder, J., R. Rajan, X. Song, A. Biedenkapp, Y. Miao, T. Eimer, B. Zhang, V. Nguyen, R. Calandra, A. Faust, *et al.* (2022). “Automated Reinforcement Learning (AutoRL): A Survey and Open Problems”. *arXiv preprint arXiv:2201.03916*.
- Pascanu, R., T. Mikolov, and Y. Bengio. (2013). “On the difficulty of training recurrent neural networks”. *ICML*.
- Peng, M., B. Zhu, and J. Jiao. (2021). “Linear Representation Meta-Reinforcement Learning for Instant Adaptation”. URL: <https://openreview.net/forum?id=LNrtNGkr-vw>.
- Perez, C., F. P. Such, and T. Karaletsos. (2020). “Generalized hidden parameter mdps: Transferable model-based rl in a handful of trials”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 04. 5403–5411.
- Polydoros, A. S. and L. Nalpantidis. (2017). “Survey of model-based reinforcement learning: Applications on robotics”. *Journal of Intelligent & Robotic Systems*. 86(2): 153–173.
- Pong, V. H., A. V. Nair, L. M. Smith, C. Huang, and S. Levine. (2022). “Offline meta-reinforcement learning with online self-supervision”. In: *International Conference on Machine Learning*. PMLR. 17811–17829.
- Prat, A. and E. Johns. (2021). “PERIL: Probabilistic Embeddings for hybrid Meta-Reinforcement and Imitation Learning”. URL: <https://openreview.net/forum?id=BIIwfP55pp>.
- Rafailov, R., V. K. Vijay, T. Yu, A. Singh, M. Phielipp, and C. Finn. (2021). “The Reflective Explorer: Online Meta-Exploration from Offline Data in Realistic Robotic Tasks”. In: *Deep RL Workshop NeurIPS 2021*.
- Raghu, A., M. Raghu, S. Bengio, and O. Vinyals. (2020). “Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML”. *ICLR*.
- Raileanu, R., M. Goldstein, A. Szlam, and R. Fergus. (2020). “Fast Adaptation via Policy-Dynamics Value Functions”. *ICML*.

- Rajendran, J., R. Lewis, V. Veeriah, H. Lee, and S. Singh. (2020). “How Should an Agent Practice?” *Proceedings of the AAAI Conference on Artificial Intelligence*. 34(04): 5454–5461. doi: [10.1609/aaai.v34i04.5995](https://doi.org/10.1609/aaai.v34i04.5995).
- Rakelly, K., A. Zhou, C. Finn, S. Levine, and D. Quillen. (2019). “Efficient off-policy meta-reinforcement learning via probabilistic context variables”. In: *International conference on machine learning*. PMLR. 5331–5340.
- Raparthy, S. C., E. Hambro, R. Kirk, M. Henaff, and R. Raileanu. (2023). “Generalization to New Sequential Decision Making Tasks with In-Context Learning”. *NeurIPS 2023 Workshop on Foundation Models for Decision Making*.
- Ravi, S. and H. Larochelle. (2017). “Optimization as a Model for Few-Shot Learning”. In: *ICLR*.
- Rechenberg, I. (1971). “Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dr.-Ing”. *PhD thesis*. Thesis, Technical University of Berlin, Department of Process Engineering.
- Ren, A. Z., B. Govil, T.-Y. Yang, K. R. Narasimhan, and A. Majumdar. (2022a). “Leveraging Language for Accelerated Learning of Tool Manipulation”. In: *6th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=nPw7jaGBrCG>.
- Ren, Z., A. Liu, Y. Liang, J. Peng, and J. Ma. (2022b). “Efficient Meta Reinforcement Learning for Preference-based Fast Adaptation”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=61UwgeIotn>.
- Rengarajan, D., S. Chaudhary, J. Kim, D. Kalathil, and S. Shakkottai. (2022). “Enhanced Meta Reinforcement Learning via Demonstrations in Sparse Reward Environments”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=kCtnkLv-W0>.

- Co-Reyes, J. D., S. Feng, G. Berseth, J. Qui, and S. Levine. (2021a). “Accelerating Online Reinforcement Learning via Model-Based Meta-Learning”. In: *Learning to Learn - Workshop at ICLR 2021*. URL: <https://openreview.net/forum?id=XCJ5PEkuMkC>.
- Co-Reyes, J. D., Y. Miao, D. Peng, E. Real, Q. V. Le, S. Levine, H. Lee, and A. Faust. (2021b). “Evolving Reinforcement Learning Algorithms”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=0XXpJ4OtjW>.
- Riemer, M., I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. (2019). “Learning to Learn without Forgetting By Maximizing Transfer and Minimizing Interference”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=B1gTShAct7>.
- Rimon, Z., A. Tamar, and G. Adler. (2022). “Meta Reinforcement Learning with Finite Training Tasks - a Density Estimation Approach”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=Y-sdZLi9R9>.
- Ritter, S., R. Faulkner, L. Sartran, A. Santoro, M. Botvinick, and D. Raposo. (2021). “Rapid Task-Solving in Novel Environments”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=F-mvpFpn_0q.
- Ritter, S., J. Wang, Z. Kurth-Nelson, S. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick. (2018a). “Been There, Done That: Meta-Learning with Episodic Recall”. In: *ICML*.
- Ritter, S., J. Wang, Z. Kurth-Nelson, S. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick. (2018b). “Been There, Done That: Meta-Learning with Episodic Recall”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. *Proceedings of Machine Learning Research*. PMLR. 4354–4363. URL: <https://proceedings.mlr.press/v80/ritter18a.html>.
- Rohani, S. R. R., S. Hedayatian, and M. S. Baghshah. (2022). “BIMRL: Brain Inspired Meta Reinforcement Learning”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 9048–9053.

- Rothfuss, J., D. Lee, I. Clavera, T. Asfour, and P. Abbeel. (2019). “ProMP: Proximal Meta-Policy Search”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=SkxXCi0qFX>.
- Sæmundsson, S., K. Hofmann, and M. P. Deisenroth. (2018). “Meta reinforcement learning with latent variable gaussian processes”. *arXiv preprint arXiv:1803.07551*.
- Salimans, T., J. Ho, X. Chen, S. Sidor, and I. Sutskever. (2017). “Evolution strategies as a scalable alternative to reinforcement learning”. *arXiv preprint arXiv:1703.03864*.
- Santoro, A., S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. (2016). “Meta-learning with memory-augmented neural networks”. In: *International conference on machine learning*. PMLR. 1842–1850.
- Schmidhuber, J. (1987). “Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook”. *PhD thesis*. Technische Universität München.
- Schmidhuber, J. (2007). “Gödel machines: Fully self-referential optimal universal self-improvers”. In: *Artificial general intelligence*. Springer. 199–226.
- Schmidhuber, J., J. Zhao, and M. Wiering. (1997). “Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement”. *Machine Learning*. 28(1): 105–130.
- Schmitt, S., M. Hessel, and K. Simonyan. (2020). “Off-policy actor-critic with shared experience replay”. In: *International Conference on Machine Learning*. PMLR. 8545–8554.
- Schoettler, G., A. Nair, J. A. Ojea, S. Levine, and E. Solowjow. (2020). “Meta-reinforcement learning for robotic industrial insertion tasks”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 9728–9735.
- Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. (2017). “Proximal policy optimization algorithms”. *arXiv preprint arXiv:1707.06347*.
- Schwartzwald, A. and N. Papanikolopoulos. (2020). “Sim-to-Real with Domain Randomization for Tumbling Robot Control”. In: *IROS*.

- Seo, Y., D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. (2022). “Masked World Models for Visual Control”. In: *6th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=Bf6on28H0Jv>.
- Seyed Ghasemipour, S. K., S. (Gu, and R. Zemel. (2019). “SMILE: Scalable Meta Inverse Reinforcement Learning through Context-Conditional Policies”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2019/file/2b8f621e9244cea5007bac8f5d50e476-Paper.pdf>.
- Shala, G., S. P. Arango, A. Biedenkapp, F. Hutter, and J. Grabocka. (2022). “AutoRL-Bench 1.0”. In: *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=RyAl60VhTcG>.
- Sharaf, A. and H. Daumé III. (2021). “Meta-Learning Effective Exploration Strategies for Contextual Bandits”. *Proceedings of the AAAI Conference on Artificial Intelligence*: 9541–9548.
- Al-Shedivat, M., T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. (2018a). “Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Sk2u1g-0->.
- Al-Shedivat, M., T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. (2018b). “Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments”. In: *International Conference on Learning Representations*.
- Shin, J., A. Hakobyan, M. Park, Y. Kim, G. Kim, and I. Yang. (2022). “Infusing model predictive control into meta-reinforcement learning for mobile robots in dynamic environments”. *IEEE Robotics and Automation Letters*. 7(4): 10065–10072.
- Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.* (2016). “Mastering the game of Go with deep neural networks and tree search”. *nature*. 529(7587): 484–489.

- Simchowitz, M., C. Tosh, A. Krishnamurthy, D. J. Hsu, T. Lykouris, M. Dudik, and R. E. Schapire. (2021). “Bayesian decision-making under misspecified priors with applications to meta-learning”. *Advances in Neural Information Processing Systems*. 34: 26382–26394.
- Singh, A., E. Jang, A. Irpan, D. Kappler, M. Dalal, S. Levine, M. Khansari, and C. Finn. (2020). “Scalable multi-task imitation learning with autonomous improvement”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2167–2173.
- Singh, S., R. L. Lewis, and A. G. Barto. (2009). “Where do rewards come from”. In: *Proceedings of the annual conference of the cognitive science society*. Cognitive Science Society. 2601–2606.
- Snell, J., K. Swersky, and R. Zemel. (2017). “Prototypical Networks for Few-shot Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett.
- Sohn, S., H. Woo, J. Choi, and H. Lee. (2020). “Meta Reinforcement Learning with Autonomous Inference of Subtask Dependencies”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=HkgsWxrtPB>.
- Song, X., W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang. (2020a). “ES-MAML: Simple Hessian-Free Meta Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=S1exA2NtDB>.
- Song, X., Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan. (2020b). “Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning”. *IROS*.
- Stadie, B. C., G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever. (2018). “Some Considerations on Learning to Explore via Meta-Reinforcement Learning”. *NeurIPS*.
- Staley, E. W., C. Ashcraft, B. Stoler, J. Markowitz, G. Vallabha, C. Ratto, and K. D. Katyal. (2021). “Meta Arcade: A Configurable Environment Suite for Meta-Learning”. *arXiv preprint arXiv:2112.00583*.
- Stone, P., G. A. Kaminka, S. Kraus, and J. S. Rosenschein. (2010). “Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination”. In: *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*.

- Subramanian, J., A. Sinha, R. Seraj, and A. Mahajan. (2022). “Approximate information state for approximate planning and reinforcement learning in partially observed systems”. *The Journal of Machine Learning Research*. 23(1): 483–565.
- Sung, F., Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. (2018). “Learning to Compare: Relation Network for Few-Shot Learning”. *CVPR*.
- Sung, F., L. Zhang, T. Xiang, T. M. Hospedales, and Y. Yang. (2017). “Learning to Learn: Meta-Critic Networks for Sample Efficient Learning”. *CoRR*. abs/1706.09529. arXiv: [1706.09529](https://arxiv.org/abs/1706.09529). URL: <http://arxiv.org/abs/1706.09529>.
- Sutskever, I. (2013). *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada.
- Sutton, R. S. (1992). “Adapting bias by gradient descent: An incremental version of delta-bar-delta”. In: *AAAI*. San Jose, CA. 171–176.
- Tack, J., J. Park, H. Lee, J. Lee, and J. Shin. (2022). “Meta-Learning with Self-Improving Momentum Target”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: https://openreview.net/forum?id=FCNMbF_TsKm.
- Tamar, A., D. Soudry, and E. Zisselman. (2022). “Regularization guarantees generalization in bayesian reinforcement learning through algorithmic stability”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 8. 8423–8431.
- Tang, Y. (2022). “Biased Gradient Estimate with Drastic Variance Reduction for Meta Reinforcement Learning”. In: *International Conference on Machine Learning*. PMLR. 21050–21075.
- Tang, Y., T. Kozuno, M. Rowland, R. Munos, and M. Valko. (2021). “Unifying gradient estimators for meta-reinforcement learning via off-policy evaluation”. *Advances in Neural Information Processing Systems*. 34: 5303–5315.
- Team, A. A., J. Bauer, K. Baumli, S. Baveja, F. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, et al. (2023). “Human-Timescale Adaptation in an Open-Ended Task Space”. *arXiv preprint arXiv:2301.07608*.

- Teh, Y., V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. (2017). “Distral: Robust multitask reinforcement learning”. *Advances in neural information processing systems*. 30.
- Tessera, K.-a. A., A. Rahman, and S. V. Albrecht. (2024). “HyperMARL: Adaptive Hypernetworks for Multi-Agent RL”.
- Thompson, W. R. (1933). “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. *Biometrika*. 25(3-4): 285–294.
- Thrun, S. and L. Pratt. (1998). “Learning to learn: Introduction and overview”. In: *Learning to learn*. Springer. 3–17.
- Todorov, E., T. Erez, and Y. Tassa. (2012). “Mujoco: A physics engine for model-based control”. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 5026–5033.
- Touati, A. and Y. Ollivier. (2021). “Learning one representation to optimize all rewards”. *Advances in Neural Information Processing Systems*. 34: 13–23.
- Triantafillou, E., T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, et al. (2020). “Meta-dataset: A dataset of datasets for learning to learn from few examples”. *International Conference on Learning Representations*.
- Tripuraneni, N., C. Jin, and M. Jordan. (2021). “Provable meta-learning of linear representations”. In: *International Conference on Machine Learning*. PMLR. 10434–10443.
- Valiant, L. G. (1984). “A theory of the learnable”. *Communications of the ACM*. 27(11): 1134–1142.
- Vanschoren, J. (2018). “Meta-learning: A survey”. *arXiv preprint arXiv:1810.03548*.
- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. (2017). “Attention Is All You Need”. *NeurIPS*.
- Vázquez-Canteli, J. R., S. Dey, G. Henze, and Z. Nagy. (2020). “CityLearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management”. *arXiv preprint arXiv:2012.10504*.

- Vázquez-Canteli, J. R. and Z. Nagy. (2019). “Reinforcement learning for demand response: A review of algorithms and modeling techniques”. *Applied energy*. 235: 1072–1089.
- Veeriah, V., M. Hessel, Z. Xu, J. Rajendran, R. L. Lewis, J. Oh, H. P. van Hasselt, D. Silver, and S. Singh. (2019). “Discovery of Useful Questions as Auxiliary Tasks”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2019/file/10ff0b5e85e5b85cc3095d431d8c08b4-Paper.pdf>.
- Veeriah, V., T. Zahavy, M. Hessel, Z. Xu, J. Oh, I. Kemaev, H. P. van Hasselt, D. Silver, and S. Singh. (2021). “Discovery of Options via Meta-Learned Subgoals”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc. 29861–29873. URL: <https://proceedings.neurips.cc/paper/2021/file/fa246d0262c3925617b0c72bb20eeb1d-Paper.pdf>.
- Vinyals, O., C. Blundell, T. Lillicrap, k. kavukcuoglu koray, and D. Wierstra. (2016). “Matching Networks for One Shot Learning”. In: *Advances in Neural Information Processing Systems*.
- Vuorio, R., J. A. Beck, G. Farquhar, J. N. Foerster, and S. Whiteson. (2021). “No DICE: An Investigation of the Bias-Variance Tradeoff in Meta-Gradients”. In: *Deep RL Workshop NeurIPS 2021*.
- Vuorio, R., S.-H. Sun, H. Hu, and J. Lim. (2019). “Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation”. In: *NeurIPS*.
- Walke, H. R., J. H. Yang, A. Yu, A. Kumar, J. Orbik, A. Singh, and S. Levine. (2022). “Don’t Start From Scratch: Leveraging Prior Data to Automate Robotic Reinforcement Learning”. In: *6th Annual Conference on Robot Learning*. URL: <https://openreview.net/forum?id=WbdaYyDkNzL>.
- Wan, M., J. Peng, and T. Gangwani. (2022). “Hindsight Foresight Relabeling for Meta-Reinforcement Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=P7OVkHEoHOZ>.

- Wang, H., J. Zhou, and X. He. (2020a). “Learning Context-aware Task Reasoning for Efficient Meta Reinforcement Learning”. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 1440–1448.
- Wang, J. X., Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick. (2018). “Prefrontal cortex as a meta-reinforcement learning system”. *Nature neuroscience*. 21(6): 860–868.
- Wang, J. X., Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. (2016). “Learning to reinforcement learn”. *arXiv preprint arXiv:1611.05763*.
- Wang, J. X., M. King, N. Porcel, Z. Kurth-Nelson, T. Zhu, C. Deck, P. Choy, M. Cassin, M. Reynolds, H. F. Song, G. Buttimore, D. P. Reichert, N. C. Rabinowitz, L. Matthey, D. Hassabis, A. Lerchner, and M. M. Botvinick. (2021). “Alchemy: A structured task distribution for meta-reinforcement learning”. *NeurIPS*.
- Wang, Q. and H. van Hoof. (2022). “Learning Expressive Meta-Representations with Mixture of Expert Neural Processes”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: <https://openreview.net/forum?id=ju38DG3sbg6>.
- Wang, Q., Y. Lv, Y. Feng, Z. Xie, and J. Huang. (2023). “A Simple Yet Effective Strategy to Robustify the Meta Learning Paradigm”. *Advances in Neural Information Processing Systems*.
- Wang, Q. and H. Van Hoof. (2022). “Model-based Meta Reinforcement Learning using Graph Structured Surrogate Models and Amortized Policy Search”. In: *Proceedings of the 39th International Conference on Machine Learning*.
- Wang, Y., Q. Yao, J. T. Kwok, and L. M. Ni. (2020b). “Generalizing from a few examples: A survey on few-shot learning”. *ACM computing surveys (csur)*. 53(3): 1–34.
- Wang, Z., Y. Zhao, P. Yu, R. Zhang, and C. Chen. (2020c). “Bayesian Meta Sampling for Fast Uncertainty Adaptation”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Bkxv90EKPB>.

- Weihs, L., U. Jain, I.-J. Liu, J. Salvador, S. Lazebnik, A. Kembhavi, and A. Schwing. (2021). “Bridging the Imitation Gap by Adaptive Insubordination”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. URL: https://openreview.net/forum?id=Wlx0DqjUTD_.
- Wen, L., S. Zhang, H. E. Tseng, B. Singh, D. Filev, and H. Peng. (2022). “Improved Robustness and Safety for Pre-Adaptation of Meta Reinforcement Learning with Prior Regularization”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 8987–8994.
- Wierstra, D., T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. (2014). “Natural evolution strategies”. *The Journal of Machine Learning Research*. 15(1): 949–980.
- Williams, R. J. (1992). “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. *Machine learning*. 8(3): 229–256.
- Woo, H., G. Yoo, and M. Yoo. (2022). “Structure Learning-Based Task Decomposition for Reinforcement Learning in Non-stationary Environments”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 8. 8657–8665.
- Wu, B., F. Xu, Z. He, A. Gupta, and P. K. Allen. (2020). “SQUIRL: Robust and Efficient Learning from Video Demonstration of Long-Horizon Robotic Manipulation Tasks”. In: *IROS*.
- Wu, Y., X. Chen, C. Wang, Y. Zhang, and K. W. Ross. (2022). “Aggressive Q-Learning with Ensembles: Achieving Both High Sample Efficiency and High Asymptotic Performance”. In: *Deep Reinforcement Learning Workshop NeurIPS 2022*. URL: <https://openreview.net/forum?id=L9MhPPvPFS>.
- Wu, Y., M. Ren, R. Liao, and R. Grosse. (2018). “Understanding Short-Horizon Bias in Stochastic Meta-Optimization”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=H1MczcgR->.
- Xian, Z., S. Lal, H.-Y. Tung, E. A. Platanios, and K. Fragkiadaki. (2021). “HyperDynamics: Meta-Learning Object and Agent Dynamics with Hypernetworks”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=pHXfe1cOmA>.

- Xie, A., A. Singh, S. Levine, and C. Finn. (2018). “Few-shot goal inference for visuomotor learning and planning”. In: *Conference on Robot Learning*. PMLR. 40–52.
- Xiong, Z., L. M. Zintgraf, J. A. Beck, R. Vuorio, and S. Whiteson. (2021). “On the Practical Consistency of Meta-Reinforcement Learning Algorithms”. In: *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=xwQgKphwhFA>.
- Xu, K., E. Ratner, A. Dragan, S. Levine, and C. Finn. (2019). “Learning a Prior over Intent via Meta-Inverse Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*.
- Xu, M., Y. Shen, S. Zhang, Y. Lu, D. Zhao, J. Tenenbaum, and C. Gan. (2022). “Prompting decision transformer for few-shot policy generalization”. In: *international conference on machine learning*. PMLR. 24631–24645.
- Xu, Z., H. P. van Hasselt, M. Hessel, J. Oh, S. Singh, and D. Silver. (2020). “Meta-Gradient Reinforcement Learning with an Objective Discovered Online”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc. 15254–15264. URL: <https://proceedings.neurips.cc/paper/2020/file/ae3d525daf92cee0003a7f2d92c34ea3-Paper.pdf>.
- Xu, Z., H. P. van Hasselt, and D. Silver. (2018). “Meta-Gradient Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2018/file/2715518c875999308842e3455eda2fe3-Paper.pdf>.
- Yan, L., D. Liu, Y. Song, and C. Yu. (2020). “Multimodal Aggregation Approach for Memory Vision-Voice Indoor Navigation with Meta-Learning”. In: *IROS*.
- Yang, J., E. Wang, R. Trivedi, T. Zhao, and H. Zha. (2022). “Adaptive Incentive Design with Multi-Agent Meta-Gradient Reinforcement Learning”. *AAMAS*.

- Yang, J., W. Hu, J. D. Lee, and S. S. Du. (2020). “Impact of representation learning in linear bandits”. *arXiv preprint arXiv:2010.06531*.
- Yang, Y., K. Caluwaerts, A. Iscen, J. Tan, and C. Finn. (2019). “NoRML: No-Reward Meta Learning”. *AAMAS*.
- Ye, H., X. Chen, L. Wang, and S. S. Du. (2023). “On the power of pre-training for generalization in rl: Provable benefits and hardness”. In: *International Conference on Machine Learning*. PMLR. 39770–39800.
- Yin, H., J. Chen, S. J. Pan, and S. Tschiatschek. (2021). “Sequential generative exploration model for partially observable reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. No. 12. 10700–10708.
- Yoon, J., T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. (2018). “Bayesian Model-Agnostic Meta-Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2018/file/e1021d43911ca2c1845910d84f40aeae-Paper.pdf>.
- Yu, L., T. Yu, C. Finn, and S. Ermon. (2019). “Meta-Inverse Reinforcement Learning with Probabilistic Context Variables”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2019/file/30de24287a6d8f07b37c716ad51623a7-Paper.pdf>.
- Yu, T., C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine. (2018). “One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning”. In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania. DOI: [10.15607/RSS.2018.XIV.002](https://doi.org/10.15607/RSS.2018.XIV.002).
- Yu, T., D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. (2020a). “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning”. In: *Conference on Robot Learning*. PMLR. 1094–1100.
- Yu, W., J. Tan, Y. Bai, E. Coumans, and S. Ha. (2020b). “Learning fast adaptation with meta strategy optimization”. *IEEE Robotics and Automation Letters*. 5(2): 2950–2957.

- Yu, W., J. Tan, C. K. Liu, and G. Turk. (2017). “Preparing for the Unknown: Learning a Universal Policy with Online System Identification”. In: *Proceedings of Robotics: Science and Systems*. Cambridge, Massachusetts. DOI: [10.15607/RSS.2017.XIII.048](https://doi.org/10.15607/RSS.2017.XIII.048).
- Yuan, H. and Z. Lu. (2022). “Robust task representations for offline meta-reinforcement learning via contrastive learning”. In: *International Conference on Machine Learning*. PMLR. 25747–25759.
- Zahavy, T., Z. Xu, V. Veeriah, M. Hessel, J. Oh, H. van Hasselt, D. Silver, and S. Singh. (2020). “A self-tuning actor-critic algorithm”. *arXiv preprint arXiv:2002.12928*.
- Zang, X., H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li. (2020). “Metalight: Value-based meta-reinforcement learning for traffic signal control”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 01. 1153–1160.
- Zhang, H. and Z. Kan. (2022). “Temporal logic guided meta q-learning of multiple tasks”. *IEEE Robotics and Automation Letters*. 7(3): 8194–8201.
- Zhang, J., J. Wang, H. Hu, T. Chen, Y. Chen, C. Fan, and C. Zhang. (2021). “MetaCURE: Meta Reinforcement Learning with Empowerment-Driven Exploration”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. *Proceedings of Machine Learning Research*. PMLR. 12600–12610. URL: <https://proceedings.mlr.press/v139/zhang21w.html>.
- Zhang, Q. and D. Chen. (2021). “A Meta-Gradient Approach to Learning Cooperative Multi-Agent Communication Topology”. In: *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*.
- Zhao, M., P. Abbeel, and S. James. (2022a). “On the Effectiveness of Fine-tuning Versus Meta-reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: https://openreview.net/forum?id=mux7gn3g_3.

- Zhao, T. Z., J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine. (2022b). “Offline meta-reinforcement learning for industrial insertion”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 6386–6393.
- Zhao, W., J. P. Queralta, and T. Westerlund. (2020). “Sim-to-real transfer in deep reinforcement learning for robotics: a survey”. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 737–744.
- Zhao, Z., A. Nagabandi, K. Rakelly, C. Finn, and S. Levine. (2021). “MELD: Meta-Reinforcement Learning from Images via Latent State Models”. In: *Conference on Robot Learning*. PMLR. 1246–1261.
- Zheng, Z., J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. Van Hasselt, D. Silver, and S. Singh. (2020). “What can learned intrinsic rewards capture?” In: *International Conference on Machine Learning*. PMLR. 11436–11446.
- Zheng, Z., J. Oh, and S. Singh. (2018). “On Learning Intrinsic Rewards for Policy Gradient Methods”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2018/file/51de85ddd068f0bc787691d356176df9-Paper.pdf>.
- Zhou, A., E. Jang, D. Kappler, A. Herzog, M. Khansari, P. Wohlhart, Y. Bai, M. Kalakrishnan, S. Levine, and C. Finn. (2020). “Watch, Try, Learn: Meta-Learning from Demonstrations and Rewards”. In: *ICLR*.
- Zhou, W., L. Pinto, and A. Gupta. (2019). “Environment Probing Interaction Policies”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=ryl8-3AcFX>.
- Zintgraf, L., S. Devlin, K. Ciosek, S. Whiteson, and K. Hofmann. (2021a). “Deep Interactive Bayesian Reinforcement Learning via Meta-Learning”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 1712–1714.
- Zintgraf, L., S. Schulze, C. Lu, L. Feng, M. Igl, K. Shiarlis, Y. Gal, K. Hofmann, and S. Whiteson. (2021b). “VariBAD: Variational Bayes-Adaptive Deep RL via Meta-Learning”. *Journal of Machine Learning Research*. 22(289): 1–39.

- Zintgraf, L., K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. (2020). “VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning”. In: *International Conference on Learning Representation (ICLR)*.
- Zintgraf, L. M., L. Feng, C. Lu, M. Igl, K. Hartikainen, K. Hofmann, and S. Whiteson. (2021c). “Exploration in Approximate Hyper-State Space for Meta Reinforcement Learning”. In: *International Conference on Machine Learning*. PMLR. 12991–13001.
- Zintgraf, L. M., K. Shiarlis, V. Kurin, K. Hofmann, and S. Whiteson. (2019). “Fast Context Adaptation via Meta-Learning”. *ICLR*.
- Zou, H., T. Ren, D. Yan, H. Su, and J. Zhu. (2021). “Learning task-distribution reward shaping with meta-learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. No. 12. 11210–11218.
- Zou, Y. and X. Lu. (2020). “Gradient-EM Bayesian Meta-Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc. 20865–20875. URL: <https://proceedings.neurips.cc/paper/2020/file/ef48e3ef07e359006f7869b04fa07f5e-Paper.pdf>.