Teachable Reinforcement Learning via Advice Distillation

Olivia Watkins
UC Berkeley
oliviawatkins@berkeley.edu

Trevor Darrell
UC Berkeley
trevor@eecs.berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Jacob Andreas
MIT
jda@mit.edu

Abhishek Gupta
UC Berkeley
abhigupta@berkeley.edu

Abstract

Training automated agents to perform complex behaviors in interactive environments is challenging: reinforcement learning requires careful hand-engineering of reward functions, imitation learning requires specialized infrastructure and access to a human expert, and learning from intermediate forms of supervision (like binary preferences) is time-consuming and provides minimal information per human intervention. Can we overcome these challenges by building agents that learn from rich, interactive feedback? We propose a new supervision paradigm for interactive learning based on "teachable" decision-making systems, which learn from structured advice provided by an external teacher. We begin by introducing a class of human-in-the-loop decision making problems in which different forms of human provided advice signals are available to the agent to guide learning. We then describe a simple policy learning algorithm that first *learns to interpret advice*, then *learns from advice* to target tasks in the absence of human supervision. In puzzle-solving, navigation, and locomotion domains, we show that agents that learn from advice can acquire new skills with significantly less human supervision required than standard reinforcement or imitation learning systems.

1 Introduction

Reinforcement learning (RL) provides a promising paradigm for building agents that can learn complex behaviors from autonomous interaction and minimal human effort. In practice, however, significant human effort is required to design and compute the reward functions that enable successful RL [48]: the reward functions underlying some of RL's most prominent success stories involve significant domain expertise and elaborate instrumentation of the agent and environment [36, 37, 43, 27, 15]. Even with this complexity, a reward is ultimately no more than a scalar indicator of how good a particular state is relative to others. Rewards provide limited information about *how* to perform tasks, and reward-driven RL agents must perform significant exploration and experimentation within an environment to learn effectively. A number of alternative paradigms for interactively learning policies have emerged as alternatives, such as imitation learning [39, 20, 49], dataset aggregation [42], preference learning [10, 6]. But these existing methods are either impractically low bandwidth (encoding little information in each human intervention) [25, 29, 10] or require costly data collection [43, 23]. It has proven challenging to develop training methods that are simultaneously expressive and efficient enough to rapidly train agents to acquire novel skills.

Human learners, by contrast, leverage numerous, rich forms of supervision: joint attention [33], physical corrections [5] and natural language instruction [9]. For human teachers, this kind of

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

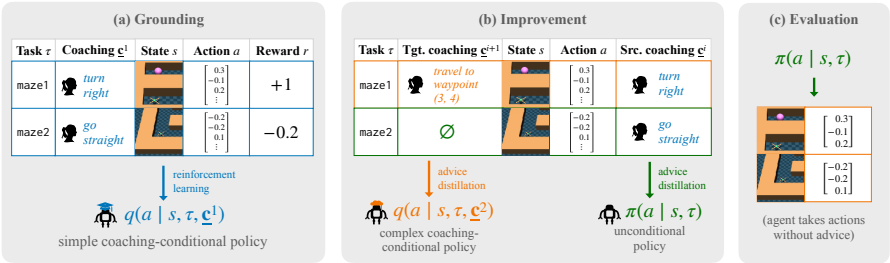

Figure 1: Three phases of teachable reinforcement learning. During the **grounding** phase (a), we train an advice-conditional policy $q(a|s, \tau, c^1)$ that can interpret a simple form of advice c^1. During the **improvement** phase (b), an external coach uses this policy to bootstrap advice-conditional policies for more complex advice forms, and ultimately an advice-independent policy $\pi(a|s, \tau)$. During the **evaluation** phase, the advice-independent policy $\pi(a|s, \tau)$ is executed to accomplish a task without the need for additional human feedback.

coaching is often no more costly to provide than scalar measures of success, but significantly more informative for learners. In this way, human learners use high-bandwidth, low-effort communication as a means to flexibly acquire new concepts or skills [45, 32]. Importantly, the interpretation of some of these feedback signals (like language), is itself learned, but can be *bootstrapped* from other forms of communication: for example, the function of gesture and attention can be learned from intrinsic rewards [38]; these in turn play a key role in language learning [30].

This paper proposes a framework for training automated agents using similarly rich interactive supervision. For instance, given an agent learning a policy to navigate and manipulate objects in a simulated multi-room object manipulation problem (in our case, BabyAI framework [8], Fig 3 left), we enable training the agent interactively using not just reward signals but advice about what actions to take ("move left"), what waypoints to move towards ("move towards $(1, 2)$"), and what sub-goals to accomplish ("pick up the yellow ball"). In doing so, we are able to ground rich channels for human supervisors to be able to direct and modify agent behavior. To actually accomplish this, we formalize a novel problem setting and supervision paradigm for interactive learning, Coaching Augmented Markov Decision Processes (CAMDPs), in which auxiliary human provided advice signals are available to the agent to guide learning. We then describe an algorithmic framework for learning in CAMDPs via alternating advice *grounding* and advice *distillation*. During the grounding phase, agents learn association between teacher-provided advice and high-value actions in the environment; during distillation, agents collect trajectories with grounded models and interactive advice, then transfer information from these trajectories to fully autonomous policies that operate without coaching. This formulation allows supervisors to guide agent behavior interactively, while enabling agents to internalize this guidance to continue performing tasks autonomously once the supervisor is no longer present. Moreover, this procedure can be extended to enable *bootstrapping* of grounded models that use increasingly sparse and abstract advice types, leveraging some types of feedback to ground others. In our experimental evaluation, we show that this procedure can allow for grounding of various different forms of advice, and this can then be used to guide the learning of new tasks up to 18x more efficiently and with less human effort needed than naïve methods for RL across puzzle-solving [8], navigation [14], and locomotion domains [8].

In summary, this paper describes: (1) a general framework (CAMDPs) for human-in-the-loop RL with rich interactive advice; (2) an algorithm for learning in CAMDPs with a single form of advice; (3) an extension of this algorithm that enables bootstrapped learning of multiple advice types; and finally (4) a set of empirical evaluations on discrete and continuous control problems in the BabyAI [8] and D4RL [14] environments demonstrating that when training on a new task, our method allows agents to converge to a higher average performance with 18x less supervision than standard RL. In doing so, we hope to introduce a new framework for human in the loop reinforcement learning that allows supervisors to use rich forms of communication to guide agent behavior acquisition.

2 Related Work

The learning problem studied in this paper belongs to a more general class of human-in-the-loop RL problems [1, 25, 29, 46, 12]. Existing frameworks like TAMER [25, 44] and COACH [29, 4] also use interactive feedback to train policies, but are restricted to scalar or binary rewards. In contrast, our work formalizes the problem of learning from arbitrarily complex feedback signals. A distinct

line of work looks to learn how to perform tasks from binary feedback with human preferences, indicating which of two trajectory snippets a human might prefer [10, 21, 46]. These techniques are only receiving a single bit of information with every human interaction, which can make the human supervision process time-consuming and tedious. In contrast, the learning algorithm we describe uses higher-bandwidth feedback signals like language subgoals or directional nudges, provided sparsely, to reduce the required effort from a supervisor.

Learning from feedback, especially provided in the form of natural language, is closely related to the instruction following in natural language processing [7, 3, 31, 40]. In instruction following problems, the goal is to produce an *instruction-conditional* policy that can generalize to new natural language specifications of behavior (at the level of either goals or action sequences [24] and held-out environments. Here, our goal is to produce an *unconditional* policy that achieves good task success autonomously—we use instruction following models to interpret interactive feedback and scaffold the learning of these autonomous policies. Moreover, the advice provided is not limited to simply providing instructions on the entire task being completed, but instead allows for more local guidance of behavior intertwined with agent execution. This provides significantly greater flexibility in altering agent behavior.

The use of language at training time to scaffold learning has been studied in several more specific settings [28]: Co-Reyes et al. [11] describe a procedure for learning to execute fixed target trajectories via interactive corrections, Andreas et al. [2] use language to produce policy *representations* useful for reinforcement learning, while Jiang et al. [22] and Hu et al. [18] use language to guide the learning of hierarchical policies. Eisenstein et al. [13] and Narasimhan et al. [34] use side information from language to communicate information about environment dynamics rather than high-value action sequences. As opposed to these settings, we aim to use interactive human in the loop advice to learn policies that can autonomously perform novel tasks in complex scenarios, even when a human supervisor is no longer present. We show how different forms of advice can be grounded interactively and can furthermore be used to ground more complex feedback forms through additional interaction.

3 Coaching Augmented Markov Decision Processes

To concretize our discussion of how to leverage rich forms of interactive human-in-the-loop supervision to modify agent behavior, we start by introducing a novel coaching-augmented MDP framework, which formalizes our approach to human-in-the-loop reinforcement learning with coaching based advice. CAMDPs build on the framework of multi-task RL and Markov decision processes (MDP), augmenting them with advice provided by a teacher in the loop through an (arbitrary) channel of communication. To situate this problem more intuitively, consider the BabyAI navigation and object repositioning environment depicted in Fig 3 left [8]. **Tasks** in this environment specify particular specific desired goal states; e.g. "place the yellow ball in the green box and the blue key in the green box" or "open all doors in the blue room". In multi-task RL, a learner's objective is produce a policy $\pi(a_t|s_t, \tau)$ that maximizes reward in expectation over tasks τ.

More formally, a **multi-task MDP** is defined by a 7-tuple $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho(s_0), \gamma, p(\tau))$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \times \tau \mapsto \mathbb{R}$ denotes the reward function, $\rho : \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ denotes the initial state distribution, $\gamma \in [0, 1]$ denotes the discount factor and $p(\tau)$ denotes the distribution over tasks. The objective in a multi-task MDP is to learn a policy π_θ that maximizes the expected sum of discounted returns in expectation over tasks: $\max_\theta J_E(\pi_\theta, p(\tau)) = \mathbb{E}_{\substack{a_t \sim \pi_\theta(\cdot|s_t,\tau) \\ \tau \sim p(\tau)}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, \tau)]$.

Why might additional supervision beyond the reward signal be useful for solving this optimization problem? Suppose the BabyAI agent in Fig 3 is in the (low-value) state shown in the figure, but could reach a high-value state by going "right and up" towards the blue key. This fact is difficult to communicate through a scalar reward, which cannot convey information about alternative actions. A side channel for providing this type of rich information at training-time would be greatly beneficial.

We formalize this as follows: in addition to the standard multi-task MDP formulation, in a **coaching-augmented MDP (CAMDP)**, training-time information, which we refer to as **advice** (or alternatively *coaching*), is provided by a teacher in the loop. We denote a teacher by $\mathcal{C} = \{\mathcal{C}^1, \mathcal{C}^2, \cdots, \mathcal{C}^i\}$, where C^j are different coaching functions each generating a different form of advice that a teacher could provide, and a teacher is simply a collection of all these coaching functions. In our empirical

evaluation we started by using a scripted, programmatic teacher for ease of evaluation but also considered an evaluation with real human supervisors as well. In every state s, the teacher chooses which coaching functions (if any) to apply, then provides advice $c^j \sim \mathcal{C}^j(s, a)$ to the agent.[1] As shown in Figure 3, advice can take many forms, for instance action advice (c^0), waypoints (c^1), language sub-goals (c^2), or any other local information relevant to task completion. Intuitively, the only real requirement for advice in this framework is that it must be prescriptive on what actions are high value to perform to make forward progress towards the high level task. Advice may be provided densely (in every state) or only infrequently.[2] Coaching in a CAMDP provides an agent local guidance on how to proceed towards solving a complex task, which may be hard to infer from a high level task description only.

As in standard reinforcement learning in an multi-task MDP, the goal in a CAMDP is to learn a policy $\pi_\theta(\cdot \mid s_t, \tau)$ that chooses an action based on Markovian state s_t and high level task information τ *without* interacting with c^j. However, we allow *learning algorithms* to use the coaching signal c^j to learn this policy more efficiently at training time (although this is unavailable during deployment). For instance, the BabyAI agent in Fig 3 can leverage hints "go left" or "move towards the blue key" to guide its exploration process but it eventually must learn how to perform the task *without* any coaching required.

Given this framework, the question becomes —how do we actually go about acquiring the coaching independent, multi-task policy $\pi_\theta(\cdot \mid s_t, \tau)$ by leveraging the human coaching signal c^j effectively at training time. We will attempt to answer these questions in the following section, followed by an experimental evaluation in Section 5

4 Leveraging Advice via Distillation

We start by formalizing the training setup we use to learn policies in CAMDPs, and then we describe an algorithm that can ground rich forms of advice and leverage this for solving new tasks.

4.1 Training Setup for Learning in CAMDPs

The challenge of learning in a CAMDP is two fold —firstly the agent must be able to interpret human in the loop advice provided via rich communication channels and use this to modify it's own behavior. This modified behavior must then be retained to perform tasks autonomously even in the absence of any human supervision. To accomplish this in the framework CAMDPs, we divide the training and evaluation process of an agent into three different phases: (1) a *grounding* phase, (2) an *improvement* phase and (3) an *evaluation* phase.

In the grounding phase, agents learn *how* to interpret advice. The result of the grounding phase is a surrogate policy $q(a_t \mid s_t, \tau, \underline{\mathbf{c}})$ that can effectively condition on advice when it is provided in the training loop. As we will discuss in Section 4.2, this phase can also make use of a *bootstrapping* process in which more complex forms of feedback are learned using signals from simpler ones. This learned surrogate policy $q(a_t \mid s_t, \tau, \underline{\mathbf{c}})$ can then be used to interpret rich forms of human in the loop advice to modify agent behavior, as we will discuss next in the improvement phase.

During the improvement phase, agents use the ability to interpret advice to learn new skills. Specifically, in an instance of the improvement phase, the learner is presented with a novel task τ_{test} that was not provided during the grounding phase, and must learn to perform this task using only a small amount of interaction in which advice c is provided by a human supervisor who is present in the loop. This advice, combined with the learned surrogate policy $q(a_t|s_t, \tau, \underline{\mathbf{c}})$, can be used to efficiently acquire an advice-*independent* policy $\pi(a_t|s_t, \tau)$, which can perform tasks without requiring any coaching.

Finally, in the evaluation phase, agent performance is evaluated on the task τ_{test} by executing the advice-independent, multi-task policy $\pi(a_t|s_t, \tau_{\text{test}})$in the environment.

[1] When only a single form of advice is available to the agent, we omit the superscript for clarity.

[2] While design of optimal coaching strategies and explicit modeling of coaches are important research topics [16], this paper assumes that the coach is fixed and not explicitly modeled.

4.2 Grounding Phase: Grounding Advice

The goal of the grounding phase is to learn a mapping from advice to contextually appropriate actions, so that advice can be used for quickly learning new tasks. In this phase, learning algorithms leverage interaction with the environment using reinforcement learning, using access to a ground truth reward function $r(s, a, \tau)$, as well as the advice $c(s, a)$ to learn a surrogate policy $q(a|s, \tau, c)$. Note that the grounding process uses privileged access to the true reward function $r(s, a, \tau)$ to learn policies $q(a|s, \tau, c)$ that can interpret advice $c(s, a)$ and r is not available in the improvement phase. The reward function r provides signal on how to interpret the advice $c(s, a)$ appropriately.

Grounding can be formulated as an ordinary multitask RL problem over a distribution of training tasks $p(\tau)$, with communication c provided as context. In order to actually perform the grounding, we can simply run standard reinforcement learning but using an advice-conditioned policy $q_\phi(a|s, \tau, \underline{\mathbf{c}})$ which has access to the advice signal $c(s, a)$ provided in the loop as context, trained to maximize the reward function $r(s, a, \tau)$. In the BabyAI environment from Fig 3, this means running RL with a policy which has access to advice forms like action advice, waypoints, or language sub-goals, and using the reward feedback to learn how to interpret these forms of advice. During this grounding process, the agent optimizes the following objective to learn how to interpret advice:

$$\max_{\phi} J(\theta) = \mathbb{E}_{\substack{\tau \sim p(\tau) \\ a_t \sim q_\phi(a_t|s_t, \tau, \underline{\mathbf{c}})}} \left[\sum_t r(s_t, a_t, \tau) \right], \tag{1}$$

The process of grounding is no different than standard multi-task RL, but including the advice $c(s, a)$ as contextual input. This formulation is simple but powerful since it makes minimal assumptions about the form of the communication and can be done with any arbitrary form of advice c. Note that since the purpose of these training environments is purely to ground communication, the tasks the agent is given during training can be much simpler than those which the agent will see at test time.

While this framework should learn to interpret advice in principle, there are a number of practical considerations that are important in training the algorithm to generalize appropriately - appropriately chosen advice representations, regularization with dropout and mutual information based regularization. We leave a thorough discussion of these to the supplementary material, instead focusing our discussion here on the conceptual framework.

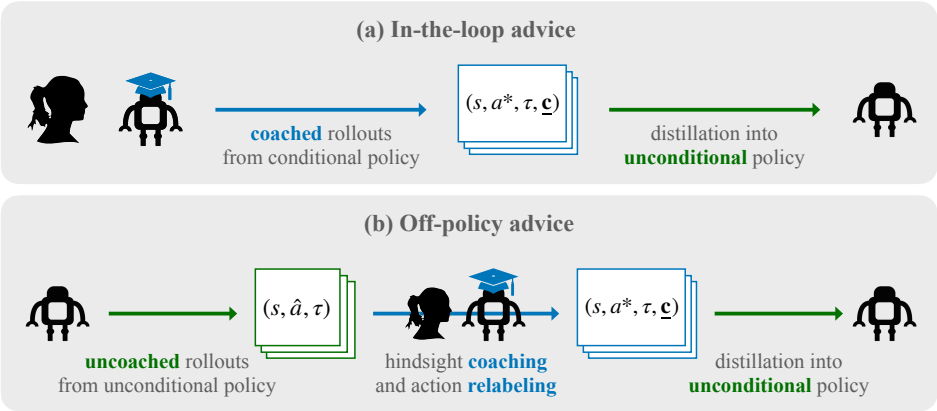

Figure 2: Illustration of the procedure of advice distillation in the on-policy and off-policy settings. During on-policy advice distillation, the advice-conditional surrogate model $q(a|s, \tau, c)$ is coached by the teacher to get optimal trajectories. These trajectories are then distilled into an *unconditional* model $\pi(a|s, \tau)$ using supervised learning. During off-policy distillation, trajectories are collected by the unconditional policy and trajectories are relabeled with advice after the fact. This same processed can be used to learn to use another advice form $q(a|s, \tau, c_i)$ rather than learning an advice-free policy.

Bootstrapping Multi-Level Advice Up until now, our formalism has largely assumed the coach only provides a single form of advice c. In practice, a coach might find it useful to use multiple forms of advice - for instance high-level language sub-goals for easy stages of the task and low-level action advice for more challenging parts of the task. While high level advice can be very informative for guiding the learning of new tasks in the improvement phase, it can often be quite difficult to ground advice forms like language sub-goals by purely doing exploration with RL. Instead of simply relying on RL to perform grounding directly from rewards r to advice c^j, we can instead bootstrap

the process of grounding one form of advice c^h from a policy $q(a|s, \tau, c^l)$ that is able to interpret a different form of advice c^l. We can use a surrogate policy which already understands (by using the grounding scheme described above) low-level advice $q(a|s, \tau, c^l)$ to bootstrap training of a surrogate policy which understands higher-level advice $q(a|s, \tau, c^h)$ by leveraging a knowledge distillation process we refer to as "bootstrap distillation". For instance, in the BabyAI domain, the agent can use its understanding of action advice to then bootstrap its understanding of language sub-goals.

Intuitively, the key idea we leverage is to use a supervisor in the loop to guide an advice-conditional policy that can interpret a low level form of advice $q_{\phi_1}(a|s, \tau, c^l)$ to perform a training task, obtaining optimal trajectories $\mathcal{D} = \{(s_0, a_0, c_0^l, c_0^h), (s_1, a_1, c_1^l, c_1^h) \cdots, (s_H, a_H, c_H^l, c_H^h)\}_{j=1}^N$ and then distilling this optimal behavior via supervised learning into a policy $q_{\phi_2}(a|s, \tau, c^h)$ that can interpret higher level advice to perform this new task **without** requiring the low level advice any longer. More specifically, we make use of an input remapping solution, as seen in Levine et al. [27], where the policy conditioned on advice c^l is used to generate optimal action labels, which are then remapped to observations with a different form of advice c^h as input. To bootstrap the understanding of an abstract form of advice c^h from a more low level one c^l, the agent optimizes the following objective to bootstrap the agent's understanding of one advice type from another:

$$\begin{aligned}\mathcal{D} =& \{(s_0, a_0, c_0^l, c_0^h), (s_1, a_1, c_1^l, c_1^h), \cdots, (s_H, a_H, c_H^l, c_H^h)\}_{j=1}^N \\ & s_0 \sim p(s_0), a_t \sim q_{\phi_1}(a_t|s_t, \tau, c^l), s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ & \max_{\phi_2} \mathbb{E}_{(s_t, a_t, c_t^h, \tau) \sim \mathcal{D}} \left[\log q_{\phi_2}(a_t|s_t, \tau, c_t^h)\right]\end{aligned}$$

4.3 Improvement Phase: Learning New Tasks Efficiently with Advice

At the end of the grounding phase, we have a well-trained advice-following agent $q_\phi(a|s, \tau, c)$ that can interpret various forms of advice. During the improvement phase, the coach introduces the agent to a new test task τ_{test} and provides advice to coach it through solving the new task. Ultimately, we want to use this advice to train a policy $\pi(a|s, \tau)$ which is able to succeed at performing the new test task τ_{test}, *without* requiring advice at evaluation time. To achieve this, we can make use of a very similar idea to the one described above for bootstrap distillation. In the improvement phase, we can leverage a supervisor in the loop to guide an advice-conditional surrogate policy $q_\phi(a|s, \tau, c)$ to perform the new task τ_{test}, obtaining optimal trajectories $\mathcal{D} = \{s_0, a_0, c_0, s_1, a_1, c_1, \cdots, s_H, a_H, c_H\}_{j=1}^N$ and then distilling this optimal behavior into an advice-independent policy $\pi_\theta(a|s, \tau)$ via supervised learning to perform this new task **without** requiring teacher in the loop advice. In doing this, we are able to learn a policy that can perform the task autonomously, but has leveraged the human advice during training as laid out in Section 3. In the BabyAI domain (shown in Fig 3 left), this improvement process would involve a teacher in the loop providing action advice or language sub-goals to the agent during learning to coach it towards successfully accomplishing a task, and then distilling this knowledge into a policy that can operate without seeing action advice or sub-goals at execution time. More formally, during the improvement phase, the agent is optimizing the following objective:

$$\begin{aligned}\mathcal{D} =& \{s_0, a_0, c_0, s_1, a_1, c_1, \cdots, s_H, a_H, c_H\}_{j=1}^N \\ & s_0 \sim p(s_0), a_t \sim q_\phi(a_t|s_t, \tau, c_t), s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ & \max_\theta \mathbb{E}_{(s_t, a_t, i) \sim \mathcal{D}} \left[\log \pi_\theta(a_t|s_t, \tau)\right]\end{aligned}$$

This improvement process, that we call advice distillation, can easily be understand in Fig 2. This distillation process is preferable over directly providing demonstrations because the advice provided can be more convenient than providing an entire demonstration (for instance, compare the difficulty of producing a demo by navigating an agent through an entire maze to providing a few waypoints). Interestingly, even if the new tasks being solved τ_{test} are quite different from the training distribution of tasks $p(\tau)$, since advice c (for instance waypoints) is provided locally and is largely invariant to this distribution shift, the agent's understanding of advice generalizes well.

4.4 Evaluation Phase: Executing tasks Without a Supervisor

In the evaluation phase, the agent simply needs to be able to perform the test tasks τ_{test} without actually requiring a supervisor in the loop. The agent's performance can be evaluated via expected

return obtained by the advice-independent agent learned in the improvement phase, $\pi(a|s,\tau)$ on the test task τ_{test} according to $J_E(\pi_\theta, p(\tau)) = \mathbb{E}_{\substack{\tau\sim p(\tau) \\ a_t\sim\pi(\cdot|s_t,\tau)}} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, \tau)]$

5 Experimental Evaluation

We aim to answer the following questions through our experimental evaluation (1) Can advice be grounded through interaction with the environment via supervisor in the loop RL? (2) Can grounded advice allow agents to learn new tasks more efficiently than standard RL? (3) Can agents bootstrap the grounding of one form of advice from another? Further details can be found at `https://sites.google.com/view/bootstrappedcoach/home`

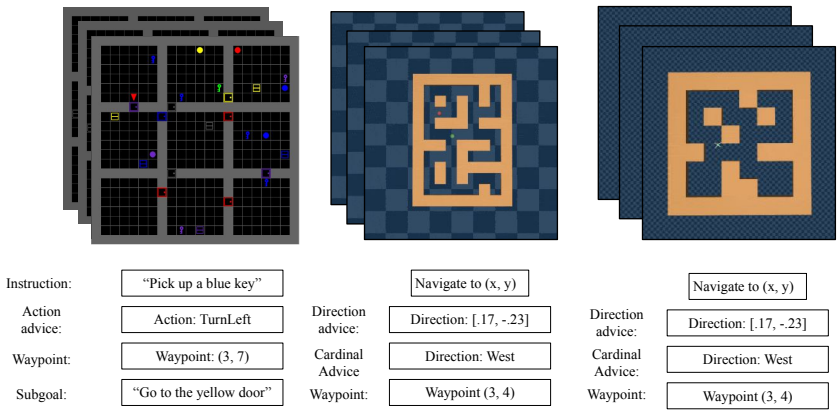

Figure 3: Evaluation Domains. (Left) BabyAI (Middle) Point Maze Navigation (Right) Ant Navigation. Each domain can have multiple different tasks to train and evaluate on. The associated task instructions are shown, as well as the types of advice available.

Evaluation Domains We evaluate on three different domains.

BabyAI: In the open-source BabyAI Chevalier-Boisvert et al. [8] gridworld environment, an agent is given tasks involving navigation, pick and place, door-opening and multi-step manipulation. We provide three types of advice:

1. **Action Advice:** the coach tells the agent the next action to take
2. **OffsetWaypoint Advice:** the coach gives the agent a tuple (x, y, b), where (x, y) is a coordinate it should visit, represented as an offset from the agent's current position, and b is a boolean telling the agent whether to interact with an object there.
3. **Subgoal Advice:** The coach gives semantic subgoals, such as "Open the blue door."

2-D Maze Navigation (PM): In the 2D navigation environment, the goal is to reach a randomly positioned target within a procedurally generated maze. We provide the agent different types of advice:

1. **Direction Advice**: The vector direction the agent should head in.
2. **Cardinal Advice**: Which of the cardinal directions (N, S, E, W) the agent should head in.
3. **Waypoint Advice**: The (x,y) position of a coordinate along the agent's route.
4. **OffsetWaypoint Advice**: The (x,y) difference between a waypoint along the agent's route and the agent's current position.

Ant-Maze Navigation (Ant): The open-source ant-maze navigation domain [14] replaces the simple point mass agent with a quadrupedal "ant" robot. The forms of advice are the same as the ones described above for the point navigation domain.

While this feedback could be provided by a human in all of these domains, in most of our experiments we use a scripted teacher in order to run experiments more efficiently.

5.1 Experimental Setup

For the environments listed above, we evaluate the ability of the agent to perform grounding efficiently on a set of training tasks, to learn new test tasks quickly via advice distillation and to leverage one form of advice to bootstrap another. The details of the exact set of training and testing tasks, as well as architecture and algorithmic details, are provided in the Appendix.

We evaluate all the environments in terms of the metric of **advice efficiency** rather than sample efficiency. By advice efficiency, we are evaluating the number of instances of teacher in the loop feedback that are needed in order to learn a task. In real-world learning tasks, this teacher is typically a human, and the cost of training largely comes from the provision of supervision (rather than time the agent spends interacting with the environment). This metric more accurately reflects the true quantity we would like to measure: the amount of human time and effort needed to provide a particular course of coaching. For simplicity, we consider every time a supervisor provides any supervision, such as a piece of advice or a scalar reward, to constitute one **advice unit** and we measure efficiency in terms of how many advice units are needed to learn a task. This simplification is definitely not true for all forms of advice, but it is challenging to design a metric which accurately captures human effort without requiring human effort each time. To validate that this advice metric seems reasonable, we perform some experiments with real humans in Section 5.5 and observe similar results. We also include plots indicating traditional sample efficiency in Appendix D.

We compare our proposed framework to an RL baseline that is provided with a task instruction but no advice. In the improvement phase, we also compare with behavioral cloning from an expert for environments where it is feasible to construct an oracle. In Appendix J we compare against alternate baseline approaches for incorporating advice.

5.2 Grounding Prescriptive Advice during Training

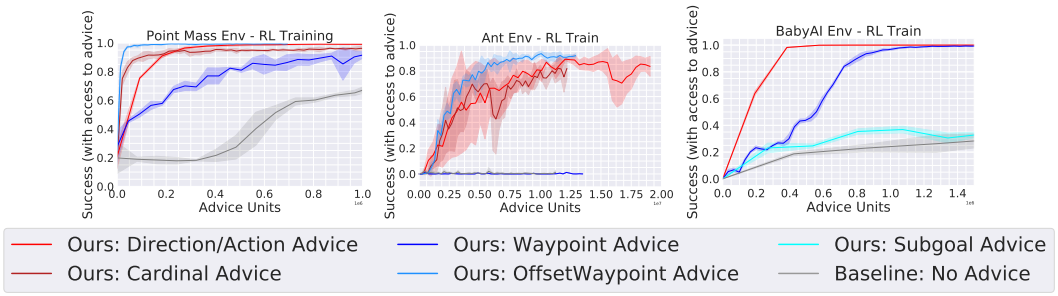

Figure 4: Performance of grounding phase as described in Section 4.2 across three domains - (left) Point Mass (PM) navigation (center) ant navigation (right two) BabyAI. All curves are trained with RL using a shaped reward on a procedurally generated set of environments. We compare the performance of an agent which conditions on high-level hints (runs in shades of blue) to one with access to low-level advice (red) to an advice-free RL baseline (gray). In most cases, providing advice speeds up learning. However, there are a few abstract advice types where RL training is slow or converges to a sub-optimal policy without bootstrapping (shown in Figure 5).

Fig 4 shows the results of the Grounding Phase, where the agent grounds advice by training an advice-conditional policy through RL. We observe the the agent learns the task more quickly when provided with advice, indicating that the agent is learning to interpret advice to complete tasks. However, we also see that the agent fails to improve much when conditioning on some more abstract forms of advice, such as Waypoint Advice in the ant environment. This indicates that the coaching form has not been grounded properly through RL. In cases like this, we instead must instead ground these advice forms through bootstrapping, as discussed in Section 4.2.

5.3 Bootstrapping Multi-Level Feedback

Once we successfully grounded the easiest form of advice, in each environment, we efficiently grounded the other forms using the bootstrapping procedure from Section 4.2. As we see in Fig 5, bootstrap distillation is able to ground new forms of advice significantly more efficiently than if we start grounding things from scratch with naïve RL. It performs exceptionally well even for advice forms where naïve RL does not succeed at all, while providing additional speed up for environments

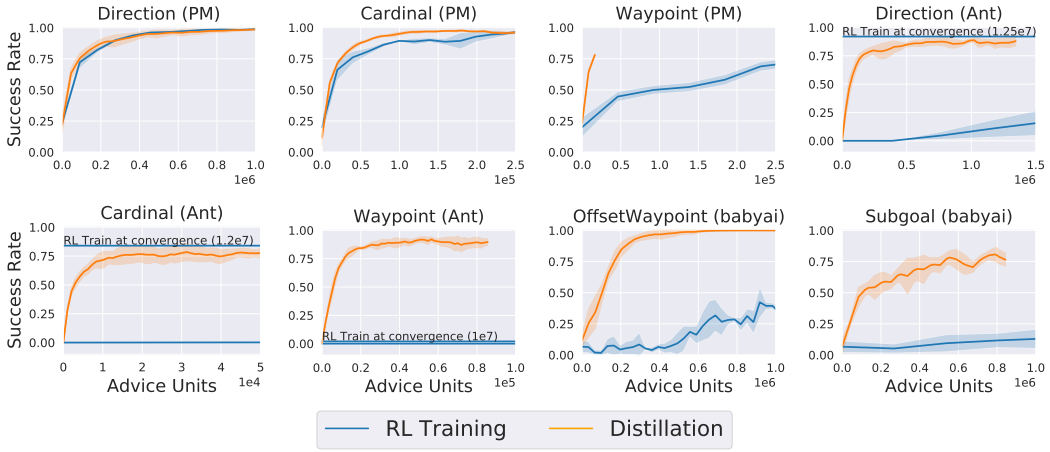

Figure 5: Performance of bootstrapping distillation procedure described in Section 4.2. Bootstrapping is able to quickly and effectively use existing grounded forms of advice (OffsetWaypoint for point navigation and Ant envs, ActionAdvice for BabyAI) to ground additional forms of advice which are harder to learn. We see that learning a new advice form through distillation is more efficient.

where it does. This suggests that advice is not just a tool to solve new tasks, but also a tool for grounding more complex forms of communication for the agent.

5.4 Learning New Tasks with Grounded Prescriptive Advice

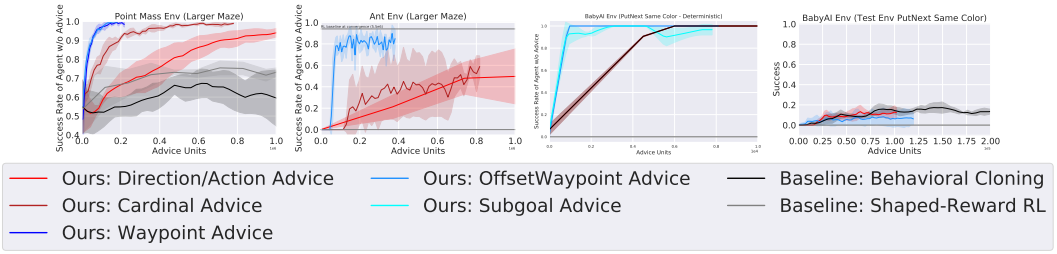

Figure 6: Learning new tasks through distillation. The agent uses an already-grounded advice channel to perform the distillation process from Section 4.3 to train an advice-free agent. Plots show the success rate of the advice-free new agent. We show that training through distillation is more advice efficient than training with RL, and comparably or more efficient than training with behavioral cloning. In general, more abstract advice forms (in blue) are more advice-efficient than than lower-level advice (in red), with occasional exceptions (far right). In the third panel note that the red ActionAdvice curve is directly below the black BC curve.

Finally, we evaluate whether we can use our grounded advice forms to guide the agent through new tasks. As we can see in Fig 6, agents which are trained through distillation from an abstract teacher on average train with less supervision than RL agents while simultaneously achieving an higher asymptotic performance. Providing high-level advice can even sometimes outperform providing demonstrations, as the high-level advice allows the human to coach the agent through a successful demostration without needing to provide an action at each timestep.

Advice grounding on the new tasks is not always perfect, however. In the rightmost panel in Figure 6, occasional errors in the advice-conditional policy's interpretation of high-advice result in it actually being more efficient to provide low-level advice than high-level advice (though both are more efficient than RL). While our scripted teacher cannot address this, a real human could dynamically switch between advice forms, as we see in our real human experiments in Section 5.5.

It's important to note here that in domains like the Ant, where standard RL fails on harder tasks (seen in Fig 4.3), and demonstrations can be difficult to provide, grounded advice provides a practical means of teaching the agent complex tasks. Interestingly, we see that as the feedback forms get more abstract, the efficiency of the agents' learning process gets better. This suggests that higher bandwidth, lower effort communication (like subgoals/waypoints) can often be extremely effective. Furthermore,

the same advice-conditional policies can be used to distill a policy in each new environment, so the up-front cost of grounding advice gets amortized over a large set of downstream tasks.

One limitation of the improvement phase as described is that the human teacher has to be continuously present as the agent is training to provide advice on every trajectory. In Appendix K we relax this by allowing the teacher to provide off-policy advice.

5.5 Real Human Experiments

To see whether our "advice unit" metric is a good proxy for human effort, we recruited people to coach agents. Advice-conditional surrogate policies were pre-trained to follow advice using a scripted teacher. Humans (authors and lab-mates) then coached these agents through solving new, more complex test environments. Afterwards, we distilled an advice-free policy from the successful trajectories. Humans provided advice through a click- and type-interface. (For instance, they could click on the screen to provide a waypoint or press a key telling the agent to drop an object.) See Fig 7.

In the BabyAI environment, we compared against a behavioral cloning baseline where the human provided per-timestep demonstrations using arrow keys. In both conditions, humans were limited to 20 mins of supervision time. Results in both conditions were high-variance, but our method shows moderate improvements. Teachers were allowed to use multiple advice forms (higher-level clicked subgoals or lower-level per-timestep actions). Anecdotally, we observed that teachers were surprised and confused when the agent failed to follow advice correctly, but they rapidly identified situations where an agent's understanding of high-level advice was imperfect and temporarily switched to providing better-grounded low-level advice in those states.

In the Ant environment, demonstrations aren't possible, and the agent does not explore well enough to learn from sparse rewards. We compare against the performance of an agent coached by a scripted teacher providing dense, shaped rewards. We see that the agent trained with 30 minutes of coaching by humans outperforms an RL agent trained with 2500 times more advice units.

6 Discussion

Summary: In this work, we introduced a new paradigm for teacher in the loop RL, which we refer to as coaching augmented MDPs. We show that CAMPDs cover a wide range of realistic scenarios and introduce a novel framework to learn how to interpret and utilize advice in CAMDPs. We show that doing so has the dual benefits of being able to learn new tasks more efficiently in terms of human effort *and* being able to bootstrap one form of advice off of another for more efficient grounding.

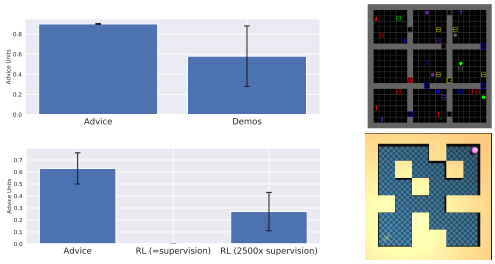

Figure 7: Top: the agent's task is to open a locked door, which involves first picking up a matching key. The agent has never seen a locked door during training. Bottom: this maze is larger than those seen in training.

Limitations: One limitation of our method is that it relies on accurate grounding of advice, which does not always happen in the presence of other correlated environment features (e.g. the advice to "open the door," and the presence of a door in front of the agent). The approaches to address this discussed in Appendix B mitigate but do not fully solve this problem. Another option for addressing this problem is to mix and match advice forms, like we did in our human experiments.

Societal impacts: As human in the loop systems such as the one described here are scaled up to real homes, privacy becomes a major concern. If we have learning systems operating around humans, sharing data and incorporating human feedback into their learning processes, they need to be careful about not divulging private information. Moreover, human in the loop systems are constantly operating around humans and need to be especially safe.

Acknowledgments: Thanks to experiment volunteers Yuqing Du, Kimin Lee, Anika Ramachandran, Philippe Hansen-Estruch, Alejandro Escontrela, and Michael Chang. Funding by NSF GRFP and DARPA's XAI, LwLL, and/or SemaFor program, as well as BAIR's industrial alliance programs.

References

[1] D. Abel, J. Salvatier, A. Stuhlmüller, and O. Evans. Agent-agnostic human-in-the-loop reinforcement learning. *CoRR*, abs/1701.04079, 2017. URL `http://arxiv.org/abs/1701.04079`.

[2] J. Andreas, D. Klein, and S. Levine. Learning with latent language. In M. A. Walker, H. Ji, and A. Stent, editors, *NAACL*, 2018.

[3] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Trans. Assoc. Comput. Linguistics*, 1:49–62, 2013. URL `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/27`.

[4] D. Arumugam, J. K. Lee, S. Saskin, and M. L. Littman. Deep reinforcement learning from policy-dependent human feedback. *CoRR*, abs/1902.04257, 2019. URL `http://arxiv.org/abs/1902.04257`.

[5] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan. Learning robot objectives from physical human interaction. In *Conference on Robot Learning (CoRL)*, 2017.

[6] D. S. Brown, W. Goo, and S. Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, 2019.

[7] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In W. Burgard and D. Roth, editors, *AAAI*, 2011.

[8] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2019.

[9] S. Chopra, M. H. Tessler, and N. D. Goodman. The first crank of the cultural ratchet: Learning and transmitting concepts through language. In A. K. Goel, C. M. Seifert, and C. Freksa, editors, *CogSci*, 2019.

[10] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *NeurIPS*, 2017.

[11] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. Andreas, J. DeNero, P. Abbeel, and S. Levine. Guiding policies with language via meta-learning. In *ICLR*, 2019.

[12] C. A. Cruz and T. Igarashi. A survey on interactive reinforcement learning: Design principles and open challenges. In R. Wakkary, K. Andersen, W. Odom, A. Desjardins, and M. G. Petersen, editors, *DIS '20: Designing Interactive Systems Conference 2020, Eindhoven, The Netherlands, July 6-10, 2020*, pages 1195–1209. ACM, 2020. doi: 10.1145/3357236.3395525. URL `https://doi.org/10.1145/3357236.3395525`.

[13] J. Eisenstein, J. Clarke, D. Goldwasser, and D. Roth. Reading to learn: Constructing features from semantic abstracts. In *EMNL*, 2009.

[14] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL `https://arxiv.org/abs/2004.07219`.

[15] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *arXiv preprint arXiv:2104.11203*, 2021.

[16] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell. Cooperative inverse reinforcement learning. *arXiv preprint arXiv:1606.03137*, 2016.

[17] D. Hejna, L. Pinto, and P. Abbeel. Hierarchically decoupled imitation for morphological transfer. In *International Conference on Machine Learning*, pages 4159–4171. PMLR, 2020.

[18] H. Hu, D. Yarats, Q. Gong, Y. Tian, and M. Lewis. Hierarchical decision making by generating and following natural language instructions. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *NeurIPS*, 2019.

[19] D. Y.-T. Hui, M. Chevalier-Boisvert, D. Bahdanau, and Y. Bengio. Babyai 1.1. *arXiv preprint arXiv:2007.12770*, 2020.

[20] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2):21:1–21:35, 2017. doi: 10.1145/3054912. URL `https://doi.org/10.1145/3054912`.

[21] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in atari. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *NeurIPS*, 2018.

[22] Y. Jiang, S. Gu, K. Murphy, and C. Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *NeurIPS*, 2019.

[23] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018. URL `http://arxiv.org/abs/1806.10293`.

[24] S. Karamcheti, E. C. Williams, D. Arumugam, M. Rhee, N. Gopalan, L. L. S. Wong, and S. Tellex. A tale of two draggns: A hybrid approach for interpreting action-oriented and goal-oriented instructions. In M. Bansal, C. Matuszek, J. Andreas, Y. Artzi, and Y. Bisk, editors, *RoboNLP@ACL*, 2017.

[25] W. B. Knox and P. Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *IEEE 7th International Conference on Development and Learning*, August 2008.

[26] H. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR, 2018.

[27] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[28] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, 2019.

[29] J. MacGlashan, M. K. Ho, R. T. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. In *ICML*, 2017.

[30] N. M. McNeil, M. W. Alibali, and J. L. Evans. The role of gesture in children's comprehension of spoken language:now they need it, now they don't. *Journal of Nonverbal Behavior*, 24 (2):131–150, 2000. doi: 10.1023/A:1006657929803. URL `https://doi.org/10.1023/A:1006657929803`.

[31] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In D. Schuurmans and M. P. Wellman, editors, *AAAI*, 2016.

[32] T. J. H. Morgan, N. T. Uomini, L. E. Rendell, L. Chouinard-Thuly, S. E. Street, H. M. Lewis, C. P. Cross, C. Evans, R. Kearney, I. de la Torre, A. Whiten, and K. N. Laland. Experimental evidence for the co-evolution of hominin tool-making teaching and language. *Nature Communications*, 6 (1):6029, 2015. doi: 10.1038/ncomms7029. URL `https://doi.org/10.1038/ncomms7029`.

[33] P. Mundy and W. Jarrold. Infant joint attention, neural networks and social cognition. *Neural Networks*, 23(8-9):985–997, 2010. doi: 10.1016/j.neunet.2010.08.009. URL `https://doi.org/10.1016/j.neunet.2010.08.009`.

[34] K. Narasimhan, R. Barzilay, and T. S. Jaakkola. Deep transfer in reinforcement learning by language grounding. *CoRR*, abs/1708.00133, 2017. URL `http://arxiv.org/abs/1708.00133`.

[35] K. Nguyen, D. Misra, R. Schapire, M. Dudík, and P. Shafto. Interactive learning from activity description. *arXiv preprint arXiv:2102.07024*, 2021.

[36] OpenAI. Openai five. *arxiv*, 2018.

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

[38] F. Poli, G. Serino, R. B. Mars, and S. Hunnius. Infants tailor their attention to maximize learning. *Science Advances*, 6(39), 2020. doi: 10.1126/sciadv.abb5053. URL `https://advances.sciencemag.org/content/6/39/eabb5053`.

[39] D. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *NeurIPS*, 1988.

[40] J. Roh, C. Paxton, A. Pronobis, A. Farhadi, and D. Fox. Conditional driving from natural language instructions. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *CoRL*, 2019.

[41] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[42] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. J. Gordon, D. B. Dunson, and M. Dudík, editors, *AISTATS*, 2011.

[43] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.

[44] G. Warnell, N. R. Waytowich, V. Lawhern, and P. Stone. Deep TAMER: interactive agent shaping in high-dimensional state spaces. In *AAAI*, 2018.

[45] S. R. Waxman and D. B. Markow. Words as invitations to form categories: evidence from 12- to 13-month-old infants. *Cogn Psychol*, 29(3):257–302, Dec 1995.

[46] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone. Leveraging human guidance for deep reinforcement learning tasks. In S. Kraus, editor, *IJCAI*, 2019.

[47] F. Zhu, Y. Zhu, X. Chang, and X. Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10012–10022, 2020.

[48] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. The ingredients of real world robotic reinforcement learning. In *International Conference on Learning Representations*, 2020.

[49] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In D. Fox and C. P. Gomes, editors, *AAAI*, 2008.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] This work does not actually use human subjects, and is largely done in simulation. But we have included a discussion in Section 6

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Math is used as a theory/formalism, but we don't make any provable claims about it.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix A for link to URL and run instructions in the README in the github repo.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All plots were created with 3 random seeds with std error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Envs we used are cited in section 5
   (b) Did you mention the license of the assets? [Yes] This is in Appendix B
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We published the code and included all environments and assets as a part of this
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We used three open source domains and collected our own data on these domains.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Environments

In all environments, at each timestep the agent receives the last unit of advice which the coach provided.

A.1 D4RL PointMass

This environment is a modified version of the environment found in the D4rl benchmark [14]. The agent 's state space consists of its own position and velocity, the target position, and a representation of the maze configuration.

The scripted coach is derived from the waypoint controller provided with the D4rl codebase. The waypoint controller finds a sequence of waypoints tracing the shortest path to the goal and computes the optimal direction the agent should head next, taking into account the next waypoint and the agent's current velocity. From this waypoint controller, we compute four advice types:

1. Direction - Optimal x-y direction to head in according to the waypoint controller.

2. Cardinal - One-hot encoding of whichever cardinal direction (N, S, E, W) has the greatest vector dot product with the optimal direction.
3. Waypoint - X-Y position of the next waypoint according to the waypoint controller.
4. OffsetWaypoint - Difference between the x-y position of the next waypoint according to the waypoint controller and the agent's current position.

Modifications from the original environment include:

1. Each reset, randomize the position of the agent's position and the goal. During training (but not test time) we also randomize maze wall configurations.
2. Modify the observation space to consist of the agent's position and velocity, the goal position, and a symbolic representation of the agent's grid. The grid is flattened and concatenated with the rest of the observation.
3. Custom semi-sparse reward provided to the agent every time it achieves an additional waypoint on the optimal path to goal.
4. Frame skip of 2.

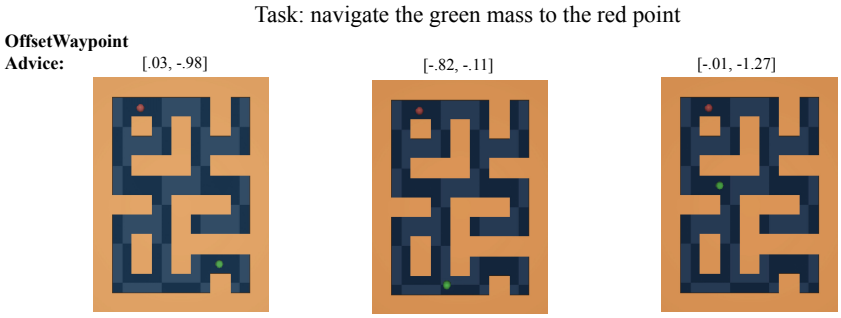

Figure 8: Example of advice offered during a trajectory in the PointMass domain with OffsetWaypoint hints.

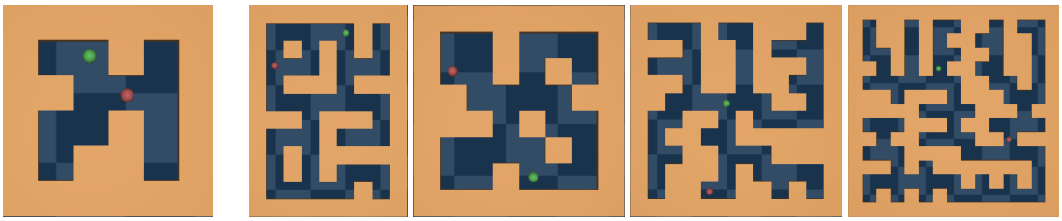

Figure 9: Left: The PointMass grounding environment consists of randomized grids of this size. Right: test environments used in the improvement phase. Results for the 2nd improvement env are reported in Figure 6, and all other improvement envs are reported in Figure 20. Tasks involve navigating to a particular position in the maze.

A.2 Ant

This environment is a modified version of the environment found in the D4rl benchmark [14]. The agent 's state space consists of the position and velocity of each of its joints, the target position, and a representation of the maze configuration. The advice forms used are identical to those in the PointMass environment. Modifications include:

1. Change the gear ratio of the ant's legs to 30.
2. Modify the observation space to consist of the agent's position, goal position, the positions and velocities of each joint, and a symbolic representation of the agent's grid.

3. Implement a custom shaped reward. The reward is the dot product between two normalized vectors: the direction the agent's torso traveled in the last timestep, and the optimal direction for the torso to travel according to the environment waypoint controller. This reward was inspired by [17]. The agent is given an additional semi-sparse reward whenever it achieves a waypoint specified by the waypoint controller.

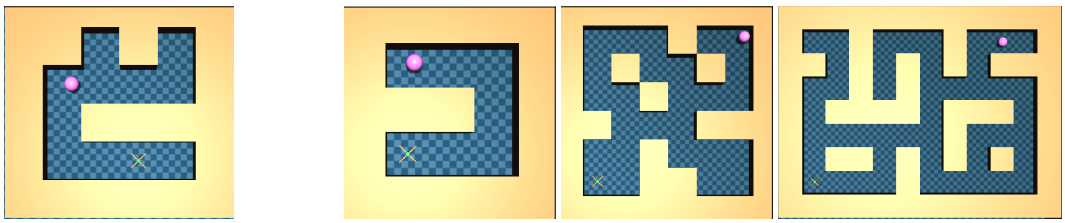

Figure 10: Left: The Ant grounding environment consists of randomized grids of this size. Right: test environments for results reported in Figure 6 (first improvement env plot) and Figure 22 (last 2 plots). Tasks involve navigating to a particular position in the maze.

A.3 BabyAI

This environment is a modified version of the environment found in the BabyAI benchmark [8]. Modifications include:

1. Make the environment fully observable.
2. Modify the observation space to be egocentric. The observation is rotated and placed within a larger padded grid such that the agent appears at the same coordinate at all times.

We use three advice types in this environment:

1. Action Advice - One-hot encoded vector specifying which discrete action to take next.
2. OffsetWaypoint Advice - X-Y coordinate offset of the location it should reach in k timesteps, where $k \sim U[2, 20]$. After k timesteps, another waypoint is sampled. The agent also receives a boolean token indicating whether it interacts with an object while reaching this waypoint. The agent also sees how many timesteps ago the advice was given.
3. Subgoal Advice - the agent is given a scripted language subgoal such as "Open the red door" or "Go to the green key."

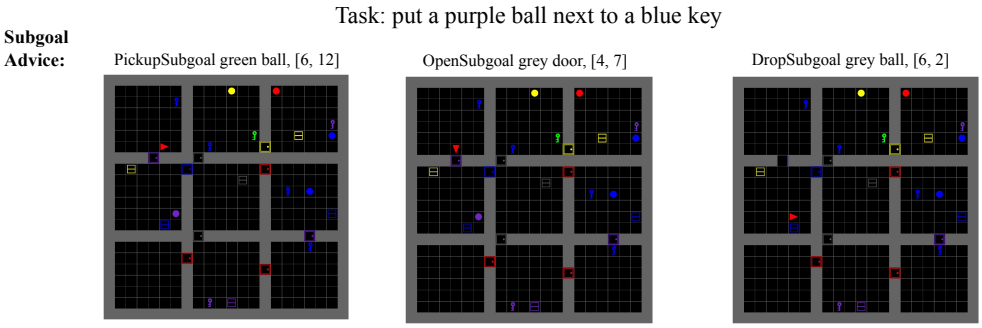

Figure 11: Example of advice offered during a trajectory in the BabyAI domain with Subgoal advice.

B Grounding Advice

We explored multiple strategies for grounding advice - i.e. making the surrogate policy rely on coaching rather than alternative features.

Figure 12: Left: BabyAI grounding environments consist of randomized tasks from the 5 levels shown. Right: test environments. The tasks are indicated by the text below the figure. The "Put Next Same Color" tasks has two variants: one with deterministic resets and one with randomized resets. All other environments have randomized resets.

B.1 Advice representation

Advice is grounded more easily when the advice representation is simple and represented in such a way that advice will appear in-distribution even within test environments. For example, consider The Waypoint and OffsetWaypoint teachers used in the PointMass and Ant environments. Since the agent's surrogate policy observes its absolute position and could compute the difference between a waypoint and its current position, both advice types should be equally informative. However, as we see in Figure 13, OffsetWaypoint hints are learned much more easily. While we are able to successfully learn a surrogate policy using Waypoint advice through bootstrapping (see Figure 5), we still see in Figure 14 that OffsetWaypoint advice generalizes better to test envs. This is likely because on test levels the Waypoint advice includes waypoint coordinates larger than ever seen during training, whereas OffsetWaypoint hints remain in-distribution.

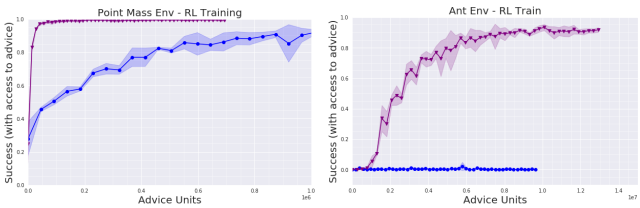

Figure 13: Advice efficiency of Waypoint advice (blue) vs OffsetWaypoint advice (purple)

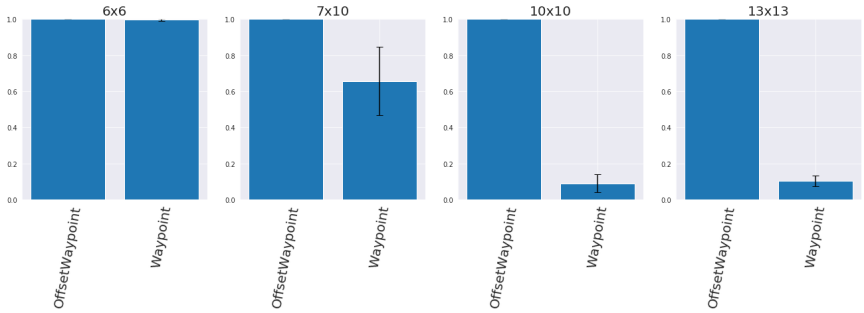

Figure 14: 0-shot success rate in new PointMass test environments for models trained with Waypoint vs OffsetWaypoint advice. OffsetWaypoint advice consistently generalizes better.

B.2 Mutual Information Loss

We implement an additional mutual information loss in the PointMass and Ant envs which rewards the agent for having high $I(c; a|s, \tau)$. As shown in Figure 15, the effect of this loss is small compared with the variation in performance across seeds.

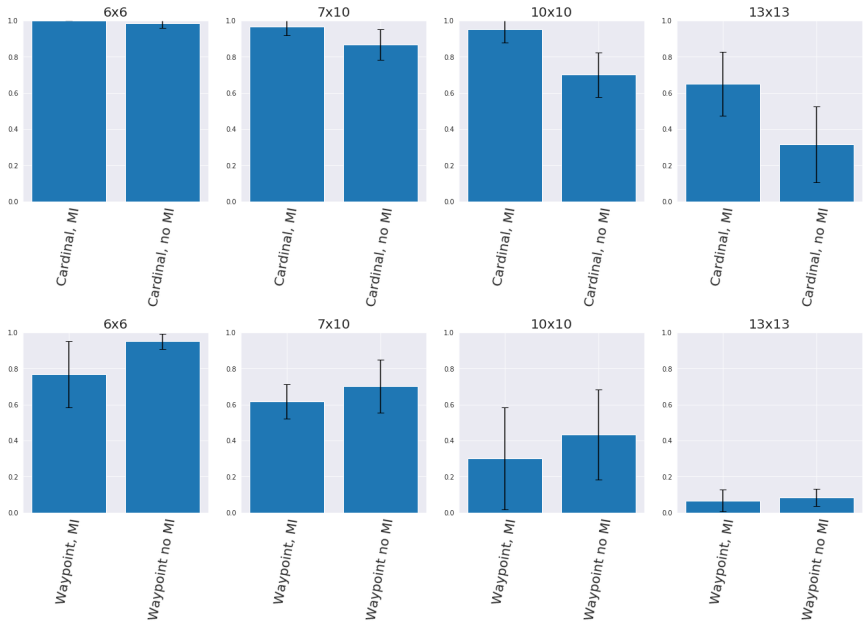

Figure 15: 0-shot success rate in new PointMass test environments for models trained with and without the mutual information auxiliary loss. We see that the mutual info has an inconsistent effect on the agent's performance.

C Code

Code can be found at `https://github.com/AliengirlLiv/teachable` under the MIT licence. The codebase incorporates elements of the meta-mb codebase, found at `https://github.com/iclavera/meta-mb` under the MIT license, the BabyAI codebase, found at `https://github.com/mila-iqia/babyai` under the BSD-3-Clause license, the d4rl codebase, found at `https://github.com/rail-berkeley/d4rl` under the Apache licence, and `https://github.com/denisyarats/pytorch_sac` which uses the MIT License.

D Sample Efficiency

Here, we report the same curves as shown in Figures 4, 5, and 6, but here we show **samples** on the x-axis rather than advice units. In all plots, one sample is one timestep. Takeaways include:

1. While low-level advice is less advice-efficient, its sample efficiency is equal or better than high-level advice.

2. For both RL training and distillation, using advice of any form is more sample efficient than using a sparse-reward baseline. Dense reward baselines are competitive during RL training, but are less sample-efficient during distillation.

3. Bootstrapping is typically more sample-efficient than RL training.

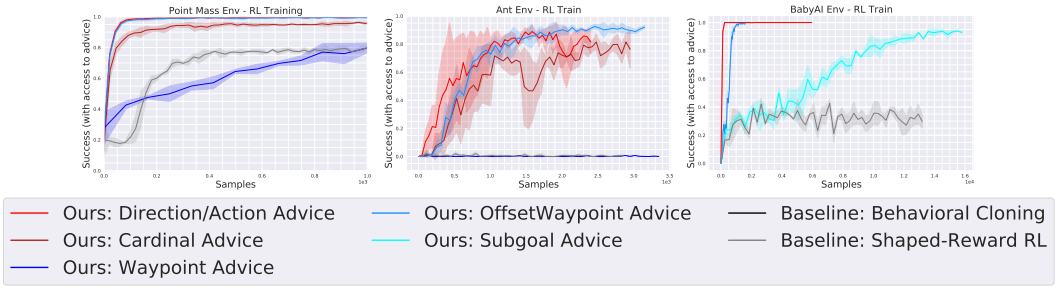

Figure 16: Plot indicating the performance in the grounding phase for various environments and forms of advice. As compared to Fig 4, this figure is plotting the x-axis as sample efficiency rather than advice efficiency.

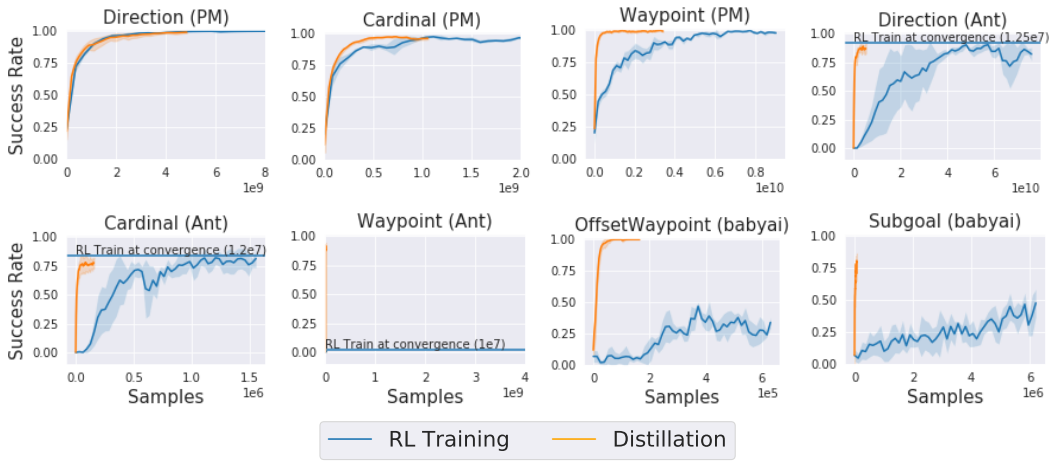

Figure 17: Plot indicating the performance in the process of bootstrapping one form of coaching from another, as shown in Fig 5. As compared to Fig 5, this figure is plotting the x-axis as sample efficiency rather than advice efficiency.

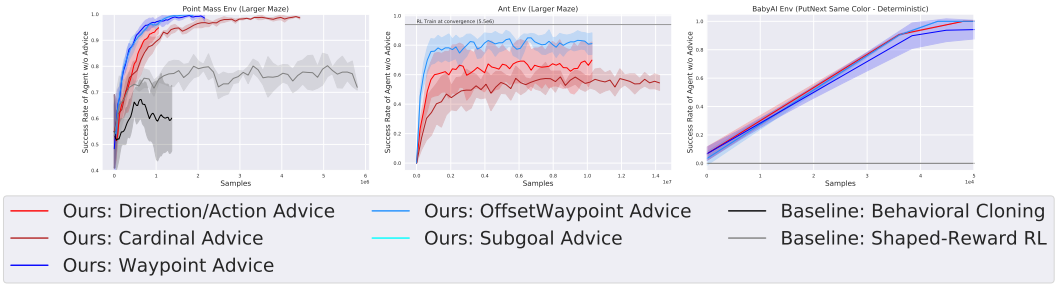

Figure 18: Plot indicating the performance in the improvement phase, demonstrating the ability to learn new tasks efficiently as shown in Fig6. As compared to Fig6, this figure is plotting the x-axis as sample efficiency rather than advice efficiency.

E Algorithm and Architecture

E.1 Algorithm

We train our our surrogate policy using using PPO as implemented in [19]. During distillation, we use behavioral cloning. Our codebase is based upon the imitation learning code from [8], with modifications to sample timesteps individually rather than as full trajectories.

E.2 Model

We build upon the architecture provided along with the BabyAI environment [19], shown with modifications in Figure 19. Modifications include removing the LSTM and incorporating advice.

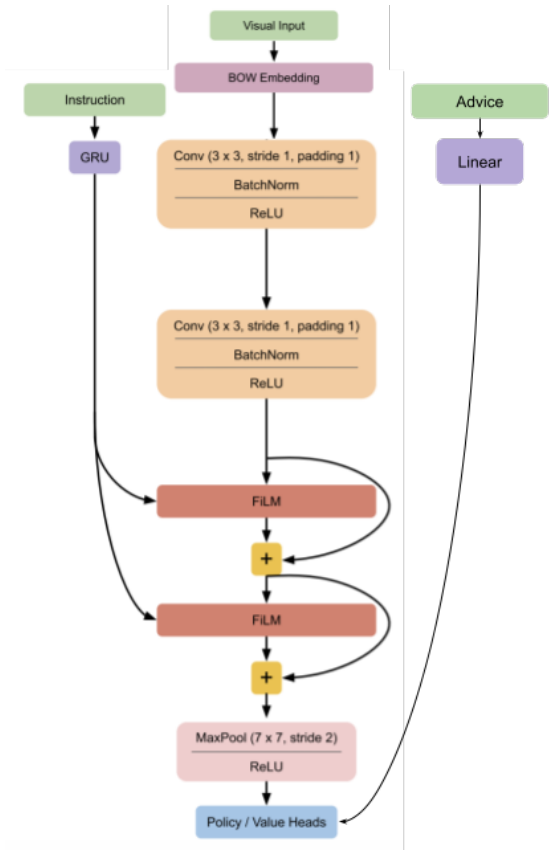

Figure 19: Architecture diagram modified from BabyAI 1.1 [19]. For the PointMass and Ant envs, which do not have image input or instructions, all model components from the conv layers through the MaxPool layer are replaced with the state input.

E.3 Hyperparameters

We chose model hyperparameters for the d4rl envs by sequentially sweeping over learning rate, batch size, control penalty coefficient, mutual info loss coefficient, entropy coefficient, discount, and update steps per iteration. We swept over three conditions: PointMass with Waypoint advice, Ant with Direction advice, and one PointMass with no advice (baseline). Similar hyperparameters worked well across both conditions, so we chose a single set of hyperparameters across all advice forms and baselines. For the BabyAI environment, since we built off the BabyAI codebase, to limit time and compute we kept the optimal model architecture for D4rl, otherwise keeping default codebase hyperparameters or hyperparameters which worked well in past experiments. For the advice reconstruction baseline, we swept over the coefficient of the advice reconstruction loss.

Hyperparameter	PM, Ant	BabyAI
Optimizer		
Adam β_1	.9	.9
Adam β_2	.999	.999
Adam ϵ	1e-5	1e-5
Adam lr	1e-3	1e-4
Model architecture		
Input embedding dim	128	128
Instr embedding dim	128	128
Advice embedding dim	128	128
Hidden layer size (actor & critic)	128	128
# hidden layers (actor & critic)	2	2
MI loss coef	.03	0
Activations	Tanh	Tanh
# film layers	2	2
PPO		
Env steps per update	4000	4000
Entropy bonus coef	1e-2	1e-4
Discount	.99	.25
Value loss coef	.5	.05
Clip ϵ	.2	.2
Grad clip	.5	.5
Batch size	200	200
GAE λ	.95	.99
Control penalty coef	.1	0
Distillation		
Batch size	2048	128
Env steps per collection batch	4000	4000
Distillation steps per collection batch	200	100
Buffer capacity	10k	1M

The codebase is under active development, and the recommended best set of hyperparameters for the newest version of the codebase will be listedin the README.

F Failure Cases

Cases in which our proposed method fails can be broken into 3 categories:

1. **Advice is not be grounded correctly.** We encountered this often. For instance, Figure 15 shows that surrogate policies conditioning on high-level advice are not perfectly able to solve tasks on 13x13 mazes. Strategies for addressing this include (1) finetuning the surrogate policy for a few iterations in the test env, either through RL or through bootstrapping from a more-successful lower-level advice form, and (2) using the strategies for improving grounding in Appendix B. Still, these methods are imperfect and imperfect grounding limits the agent's ability to generalize to arbitrary new tasks.
2. **Test-time tasks cannot be solved easily using previously-grounded advice.** Some test-time task might not be expressible in terms of high-level advice (e.g. subgoals) the agent understands. However, the agent can still be coached to success on this task using lower-level Action Advice. Future extensions to this work will involve providing abstract advice during easy portions of the task, but dropping down to lower-level advice during portions of the task where the agent isn't able to follow high-level advice.
3. **The surrogate policy succeeds, but distillation fails.** Since the trajectories obtained by running the surrogate policy can be thought of as expert demos, and we perform distillation through behavioral cloning, distillation fails in environments in which behavioral cloning from expert data fails. If this failure is due to compounding error, the distillation step could be performed using DAgger [42], since the surrogate policy can be used as an interactive expert. We found this wasn't necessary in the environments we tested on.

G Compute

The experiments in this paper were run on 8 11019MiB GPUs for about 2.5 weeks.

H Additional Plots

Here, we include additional distillation plots which there was not space for in the main paper. In all plots, if an agent was not able to solve a test env 0-shot, the advice-conditioned surrogate policy was first finetuned in the test environment for a few (about 12) iterations. The advice collected during finetuning is included in the x-axis of these plots.

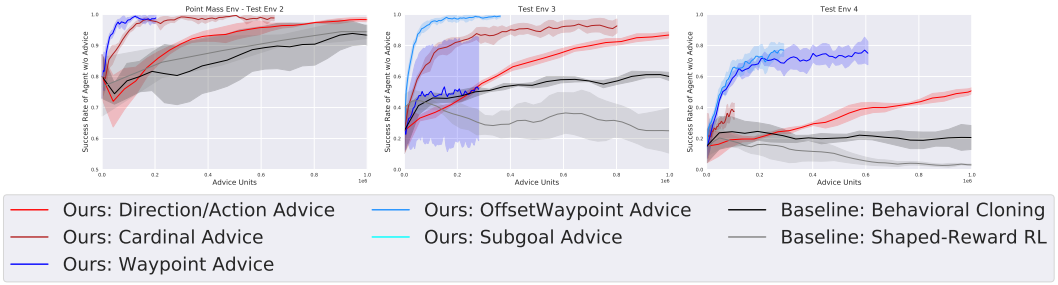

Figure 20: Figure indicating performance during the improvement phase for the pointmass environment on additional testing levels.

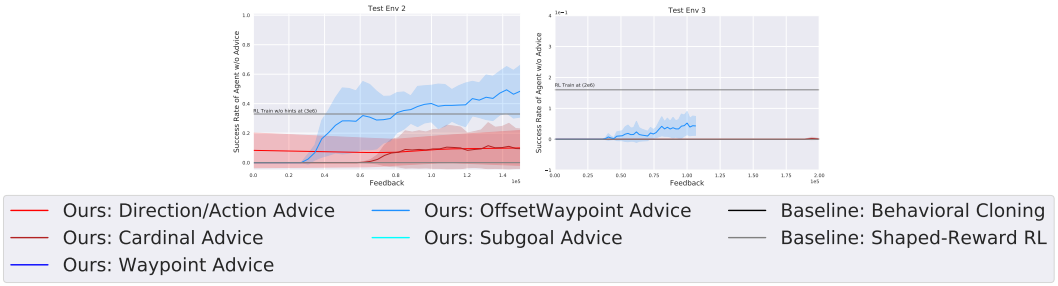

Figure 21: Figure indicating performance during the improvement phase for the Ant environment on additional testing levels.

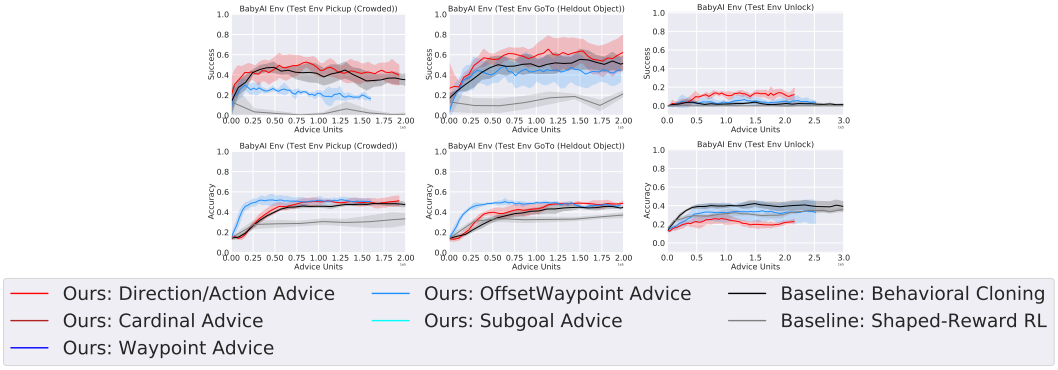

Figure 22: Figure indicating performance during the improvement phase for several BabyAI environments on additional testing levels. Bottom: providing high-level advice improves the speed at which the policy's per-timestep accuracy reaches convergence (accuracy is taken with respect to a scripted oracle). Top: this increased accuracy does not translate into higher success, likely because of all the places in the environment where the agent can get stuck, resulting in a failed trajectory even with a generally-accurate policy..

I Robustness to Noise

Unlike in the real-human experiments, where advice provided is often noisy, the advice in most of our simulated experiments never makes mistakes. We evaluated whether our method can be used to coach agents when the simulated advice is noisy by introducing noise into the improvement phase. OffsetWaypoint advice was provided by a scripted teacher which gridifies the maze and plots a path through the grid. The noise introduced randomly replaced a certain fraction of waypoints with points sampled from adjacent grid cells. Incorrect waypoints were provided for 5 timesteps at a time. Results are reported in Figure 23, where we see that the agent is able to learn quickly with up to 50% noise in the provided samples.

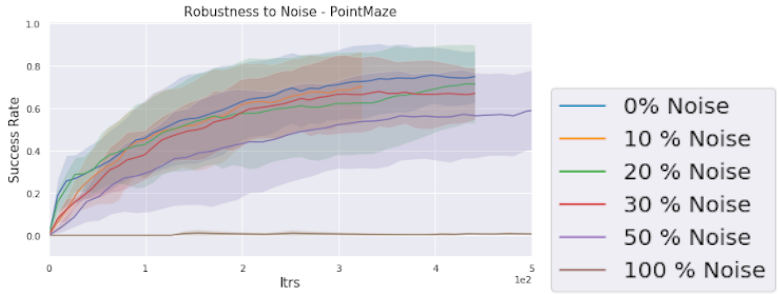

Figure 23: Performance on the improvement phase in the point maze environment. The noise percentage refers to the percent of advice samples which were replaced by adjacent incorrect waypoints.

J Alternative Ways to Use Advice

We explored several alternative ways to provide advice, but ultimately found the approach presented in our works most reliably.

J.1 Advice Reconstruction

Rather than providing advice as an input to the agent's observation, we incorporated advice by adding an auxiliary loss to predict it, similar to [47]. Figure 24 shows the effect of advice reconstruction in the training envs. While we found this improved performance over a pure RL baseline, we found the advice wasn't able to speed up learning in challenging environments like Ant.

Figure 24: We compare the effect of providing the advice in the agent's action space against a baseline of using the advice as a reconstruction loss. We find that by passing in the advice in the observation we get sped-up learning in challenginv environments like ant and BabyAI, whereas we get no learning speedup from the auxiliary loss.

J.2 Hindsight Relabeling

Rather than provide prescriptive advice, we explored having the teacher provide advice by relabeling an agent's trajectory with the goal it achieved. We can then train this now-successful relabeled trajectory using supervised learning, as was done in [35]. However, we found that hindsight relabeling performed poorly, except on the simplest environments. However, we only tried a very naive approach to getting this method to work, and it's possible more sophisticated methods could succeed here.

J.3 Hierarchical RL

We explored an alternate method of using advice using hierarchical rl. We modified the grounding phase to train an advice-conditional policy as described in our paper, but also do supervised training of an advice generation high-level policy which takes in the state and task embedding and outputs advice. During the improvement phase, the teacher directly provides advice to the low-level policy to coach the agent to success on the new task. Simultaneously, we can fine-tune the high-level policy on advice from this environment. (No rewards or low-level supervision is provided during this phase.) At evaluation time, the advice generation high-level policy generates advice, which the low-level policy executes. Results using this approach are shown in Fig J.3, where we see that it performs comparably to our approach (labeled "Distill Flat") across a range of advice types and conditions. However, we only show results on a few simple environments and advice types. With more complex advice representations (e.g. waypoints, subgoals), we found we were not able to even learn a low-level policy which could predict advice well enough to succeed on the train levels, much less on the test environments reported in Fig J.3.

Env	Advice	Distill Flat	Finetune Hierarchical
PointMaze Test Env 1	Direction	0.98 ± 0.01	0.99 ± 0.0
PointMaze Test Env 1	Cardinal	0.34 ± 0.39	0.27 ± 0.32
PointMaze Test Env 2	Direction	0.91 ± 0.03	0.91 ± 0.02
PointMaze Test Env 2	Cardinal	0.21 ± 0.25	0.21 ± 0.25
PointMaze Test Env 2	OffsetWaypoint	0.97 ± 0.02	0.95 ± 0.0
PointMaze Test Env 3	Direction	0.84 ± 0.05	0.94 ± 0.04
PointMaze Test Env 3	Cardinal	$.2 \pm 0.24$	0.17 ± 0.21
PointMaze Test Env 3	OffsetWaypoint	0.96 ± 0.02	0.94 ± 0.04

Figure 25: Success rate of the distillation phase using our method vs the hierarchical RL method. Typically, these methods perform at approximately the same rate. However, these test environment evaluation were only done for advice forms where the agent was able to learn a decent advice predictor on the train environments in the first place.

K Learning New Tasks from Off-Policy Data

One limitation of the improvement phase as described is that the human teacher has to be continuously present as the agent is training to provide advice on every trajectory. We relaxed this requirement by providing the advice in hindsight rather than in-the-loop, as shown in Figure 2. Concretely, we roll out our (initially untrained) advice-free policy. After the fact, the teacher provides high-level advice at a few points along each trajectory. Next, we use the advice-conditional surrogate policy to provide near-optimal actions for each timestep of this newly advice-labeled trajectory. This allows us to use behavioral cloning to train the advice-free agent on this trajectory. This process can be though of as the teacher performing DAgger [41] at the level of high-level advice (as was done in in [26]) rather than low-level actions. Results are shown in Figure 26. This DAgger-like scheme for soliciting advice performs comparably to receiving real-time advice but removes the need for a human to be constantly present in the loop. It also opens avenues for future work to explore using active learning techniques to label only the most informative trajectories.

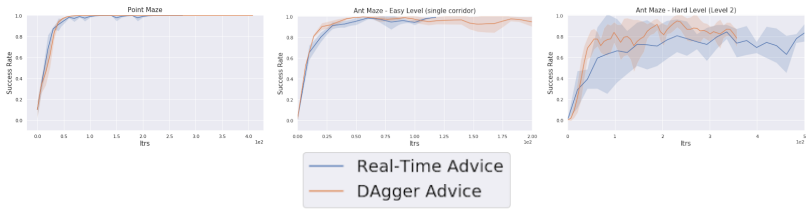

Figure 26: All curves show the success rate of an advice-free policy trained via distillation from an advice-conditional surrogate policy. We used the same advice form in all curves, so advice-efficiency results look very similar to the sample efficiency results reported here. All curves use the OffsetWaypoint advice form, and results are averaged over three seeds.