Teachable Reinforcement Learning via Advice Distillation

Anonymous Author(s)
Affiliation
Address
`email`

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 low-bandwidth. 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 that 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* how to autonomously complete a target task. In a set of puzzle-solving, navigation, and locomotion domains, we show that agents that learn from advice can acquire new skills with significantly less advice supervision than ordinary reinforcement or imitation learning.

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 [43]: the reward functions underlying some of the most prominent success stories in RL involve significant domain expertise and elaborate instrumentation of the agent and environment [32, 33, 38, 24, 15]. Despite this complexity, reward functions provide relatively little information to learners: as rewards are simply scalars indicating how good a particular state is relative to others, they provide limited information about *how* to perform tasks. 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 [35, 18, 44], dataset aggregation [37], and preference learning [10, 6] that can learn in the absence of rewards. However, existing methods are either impractically low bandwidth [23, 26, 10] or require costly data collection [38, 21].

Human learners, by contrast, are not confined to scalar evaluations of behavior when learning new tasks. Instead, they are able to leverage numerous, richer forms of supervision: joint attention [30], physical nudges [5] and natural language instruction [9]. For human teachers, this kind of 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 [40, 29]. 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 [34]; these in turn play a key role in language learning [27].

This paper proposes a framework for training automated agents using similarly rich interactive supervision. For instance, given an agent learning a navigation policy in the BabyAI framework [8] (Fig 1), we enable training the agent interactively using not just reward signals but action advice ("move left"), waypoints ("move towards $(1, 2)$") and sub-goals ("pick up the yellow ball"). To do so, we formalize a novel problem setting and supervision paradigm for interactive learning, Coaching Augmented Markov Decision Processes (CAMDPs), in which auxiliary advice signals are available to the agent to guide learning. We then describe an algorithmic framework for learning in CAMDPs via sequencing *grounding* and *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 procedure can be extended to enable *bootstrapping* of grounded models that use increasingly sparse and abstract advice types. 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].

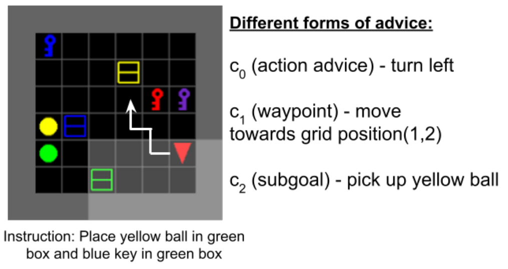
Different forms of advice:
c_0 (action advice) - turn left
c_1 (waypoint) - move towards grid position(1,2)
c_2 (subgoal) - pick up yellow ball
Instruction: Place yellow ball in green box and blue key in green box

Figure 1: Example in the BabyAI [8] domain, introducing the CAMDP problem setting. The agent is provided with an instruction qualifying which task has to be solved, and is also provided with various forms of advice as shown here - action advice, waypoints and language subgoals. The agent can use the advice c^0, c^1, c^2 to quickly accomplish new tasks.

In summary, this paper describes: (1) a general framework (CAMDPs) for teacher-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 with a given advice budget, our method allows agents to converge to a higher average performance with 18x less supervision than standard RL.

2 Related Work

The learning problem studied in this paper belongs to a more general class of human-in-the-loop RL problems [1, 23, 26, 41, 12]. Existing frameworks like TAMER [23, 39] and COACH [26, 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, 19, 41]. 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, 28, 36]. 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 [22] 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.

The use of language at training time to scaffold learning of language-unconditional behaviors has been studied in several more specific settings [25]: 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. [20] and Hu et al. [17] use language to guide the learning of hierarchical policies. Eisenstein et al. [13] and Narasimhan et al. [31] use side information from language to communicate information about environment dynamics rather than high-value action sequences.

3 Coaching Augmented Markov Decision Processes

We start by introducing our 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 navigation and object repositioning environment depicted in Fig 1 [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, **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 agent in Fig 1 is in the (low-value) state shown in the figure, but could reach a high-value state by going "left and up". This fact is difficult to communicate through a scalar reward (which cannot convey information about alternative actions) or through a state observation (since we ultimately wish to produce an agent that can act on its own, without hints in the observation space). A side channel for training-time information 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. While the teacher could be a human supervisor, in our empirical evaluation we used a scripted teacher for ease of evaluation. In every state, the teacher chooses which coaching functions (if any) to apply in a given state, then provides advice $c^j \sim \mathcal{C}^j(s, a)$ to the agent.[1] As shown in Figure 1, advice can take many forms: action advice (c^0), waypoints (c^1), language sub-goals (c^2), or any other local information relevant to task completion. Advice may be provided densely (in every state) or only infrequently.[2]

As in ordinary MDP learning, 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 train this policy more efficiently. For instance, the agent in Fig 1 can leverage hints "go left" or "move towards the blue square" to guide its exploration process but it eventually must learn how to perform the task *without* any coaching required.

4 Leveraging Advice via Distillation

We start by formalizing the training setup we use to solve CAMDPs, and then we describe a practical algorithm to actually be able to ground and leverage advice for solving new tasks.

[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.

4.1 Training Setup for Learning in CAMDPs

In order to effectively leverage advice in 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 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.

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 is provided. This advice, combined with the learned surrogate policy $q(a_t|s_t, \tau, \underline{\mathbf{c}})$, is used to learn 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, instruction conditional policy $\pi(a_t|s_t, \tau_{\text{test}})$in the environment.

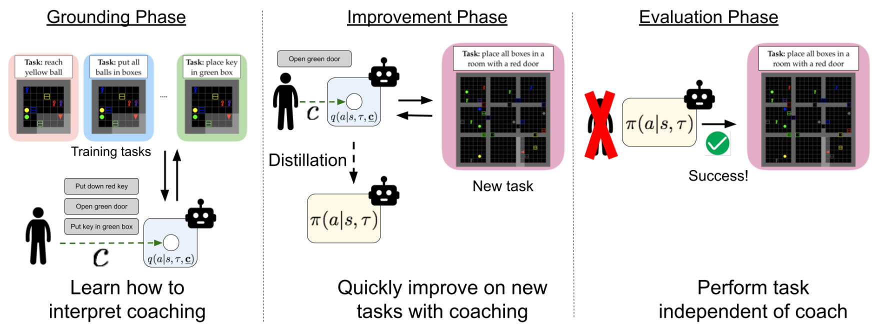
Grounding Phase
Improvement Phase
Evaluation Phase
Task: reach yellow ball
Task: put all balls in boxes
Task: place key in green box
Training tasks
Put down red key
Open green door
Put key in green box
c
$q(a|s,\tau,\underline{\mathbf{c}})$
Learn how to interpret coaching
Open green door
Distillation
$\pi(a|s,\tau)$
Task: place all boxes in a room with a red door
New task
Quickly improve on new tasks with coaching
Success!
Perform task independent of coach

Figure 2: Figure illustrating the three phases of learning a coachable system - (1) grounding (2) improvement and (3) evaluation. During the grounding phase, the goal is to learn a surrogate policy $q(a|s, \tau, c)$ that can interpret advice c. During the improvement phase, the goal is to use the surrogate policy and a coach to quickly learn new tasks and enable an advice-independent policy $\pi(a|s, \tau)$ to solve the task. During the evaluation phase, the advice-independent policy $\pi(a|s, \tau)$ is executed to accomplish a task.

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 the 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 $c(s, a)$.

Grounding can be formulated as an ordinary multitask RL problem with a distribution of training tasks $p(\tau)$. Since the function 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. 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, trying to maximize the reward function $r(s, a, \tau)$. In the example from Fig 1, 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:

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

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 describe these in the Appendix.

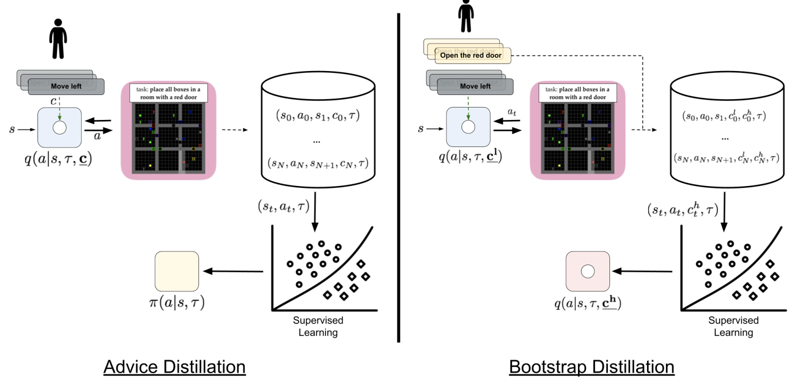
Move left
task: place all boxes in a room with a red door
$(s_0, a_0, s_1, c_0, \tau)$
...
$(s_N, a_N, s_{N+1}, c_N, \tau)$
$q(a|s, \tau, \underline{c})$
(s_t, a_t, τ)
$\pi(a|s, \tau)$
Supervised Learning
Advice Distillation
Open the red door
$(s_0, a_0, s_1, c_0^l, c_0^h, \tau)$
$(s_N, a_N, s_{N+1}, c_N^l, c_N^h, \tau)$
$q(a|s, \tau, \underline{c^l})$
(s_t, a_t, c_t^h, τ)
$q(a|s, \tau, \underline{c^h})$
Supervised Learning
Bootstrap Distillation

Figure 3: Illustration of the procedure of advice distillation (left) and bootstrap distillation (right). During 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 bootstrap distillation a similar process is followed, but the distillation is from one form of advice c^l to another c^h to learn a surrogate policy $q(a|s, \tau, c^h)$

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 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 Fig 1, the agent can use its understanding of action advice to then bootstrap it's understanding of language sub-goals.

Intuitively, the key idea we leverage is to use a supervisor in the loop to guide a 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 the input remapping trick, seen in Levine et al. [24], 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 a 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 a advice-independent policy $\pi_\theta(a|s, \tau)$ via supervised learning to perform this new task **without** requiring teacher in the loop advice. In the example from Fig 1, 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:

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

This improvement process, that we call advice distillation, can easily be understand in Fig 3. 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 $J_E(\pi_\theta, p(\tau)) = \mathbb{E}_{\substack{\tau\sim p(\tau) \\ a_t\sim\pi(\cdot|s_t,\tau)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, \tau)\right]$

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 the learned grounding 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`

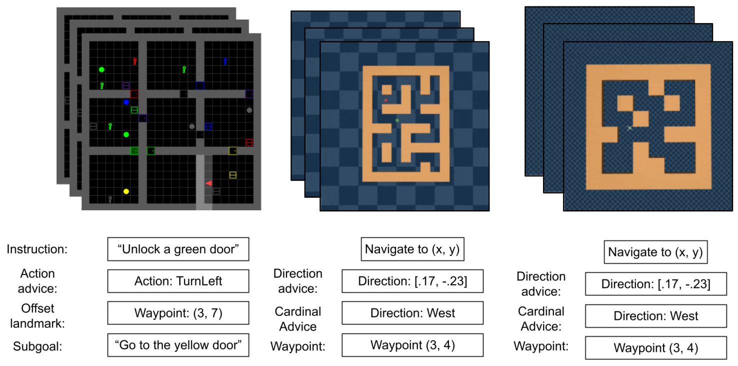

Figure 4: Evaluation Domains. (Left) BabyAI (Middle) Point Navigation (Right) Ant Navigation. Each domain can have multiple different tasks to train and evaluate on. The associated task instructions are shown, as well as the different types of advice that are available for each domain.

Evaluation Domains In our work, we consider evaluation on 3 different domains.

BabyAI: The first domain, which we refer to as BabyAI, is based on the open-source visual navigation and object manipulation domain introduced by Chevalier-Boisvert et al. [8]. This environment involves picking up objects with different characteristics, and then placing them at different positions. The family of tasks in BabyAI involves various navigation, pick and place, door-opening and multi-step manipulation tasks identified by an instruction. 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 the position it should visit along its path, represented as an offset from the agent's current position, and b is a boolean telling the agent whether to interact with an object at that coordinate.
3. **Subgoal Advice:** The coach gives semantic subgoals, such as "Find the blue key."

2-D Maze Navigation (PM): The second domain we consider is an open-source 2 dimensional maze navigation problem [14]. This task involves navigating a complex maze with a 2 dimensional point robot, where the start positions, the goal positions and the maze configuration are randomly varied across tasks. The agent is provided with access to its own position, the target position and a representation of the maze configuration. 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 third domain we consider involves scaling up to higher dimensional state and action spaces and solving a more challenging control problem. In the ant-maze navigation domain (from open source [14]), the goal is to navigate around a complex maze as shown in Fig 4, but with the agent controlling a quadrupedal "ant" robot, rather than a point. The forms of advice are exactly the same as the ones described above for the point navigation domains.

While this feedback could be provided by a human in all of these domains, in our work we make use of a scripted teacher to generate all the advice as a proxy to be able to run computational studies more efficiently. The details of each scripted teacher are provided in the Appendix.

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 tasks for grounding advice in Fig 5 and the set of testing tasks to learn quickly in Fig 7, as well as model classes, particular choice of RL algorithm and success metrics and hyperparameters 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 ability of an agent to learn a task efficiently in terms of *teacher* feedback. 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. We also include plots indicating traditional sample efficiency in the appendix.

We compare our proposed framework to an RL baseline that is provided with a task instruction but no advice. We also compare with behavioral cloning (imitation learning) from an expert for environments where it is feasible to construct an oracle. Inspired by [42], we also consider a baseline where rather than conditioning on advice the agent's model is trained to predict the advice as an auxiliary loss.

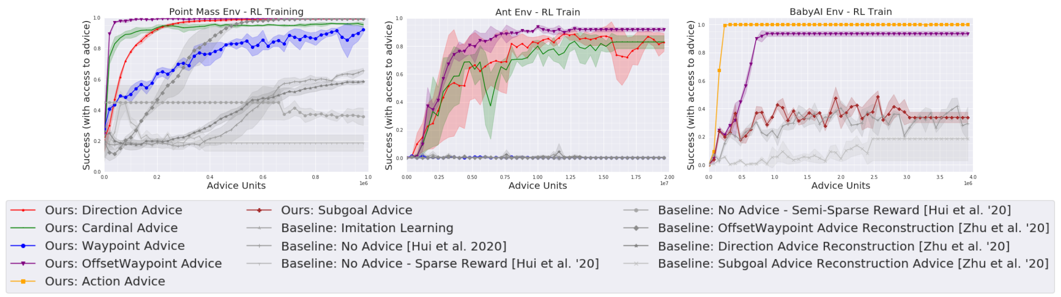

Figure 5: Performance of grounding phase as described in Section 4.2 across three domains - (left) Point Mass (PM) navigation (center) ant navigation (right) BabyAI. All curves are trained with RL using a dense reward unless otherwise specified and are both trained and evaluated on a procedurally generated set of environment configurations. We compare the performance of an agent which conditions on various hint forms (colored runs) to various baselines (gray). We see that the advice efficiency of our method typically outperforms baseline methods. 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 6).

5.2 Grounding Prescriptive Advice during Training

Fig 5 shows the results of the Grounding Phase, where the agent is learning an advice-conditioned policy. The agent is able to learn more quickly when provided with advice than when it is not, showing that advice actually is enabling the agent to learn faster than it does on its own by providing it more directed and localized guidance about how to perform tasks. Note that the learning process here is both grounding the interpretation of advice and implicitly using this interpretation to learn the training tasks more efficiently. However, we also see that some more abstract forms of advice, such as Waypoint Advice in the ant environment, cannot easily be grounded through RL as they are more abstract and require more directed exploration to be grounded. This motivates the need for a bootstrapping approach to learning abstract feedback, as discussed in Section 4.2.

5.3 Bootstrapping Multi-Level Feedback

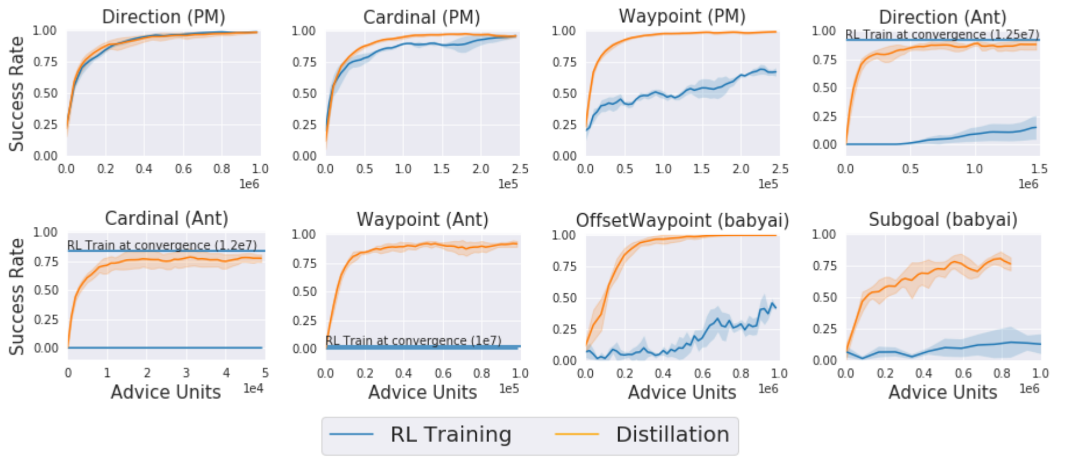

Figure 6: Performance of bootstrapping distillation procedure described in Section 4.2. Bootstrapping is able to quickly and effectively use existing grounded forms of advice (OffsetWaypoint for point navigation and Ant envs, Action for BabyAI) to help with the grounding process of additional forms of advice which are harder to learn. We see that learning a new advice form through distillation is more efficient than using RL in many cases.

Secondly, we consider the ability of our method to bootstrap multiple different forms of advice off of each other We first ground a single form of advice c^l on the training tasks, then use the bootstrap distillation scheme from Section 4.2 to ground an additional form of advice on the training tasks c^h.

As can be seen from Fig 6, 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 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

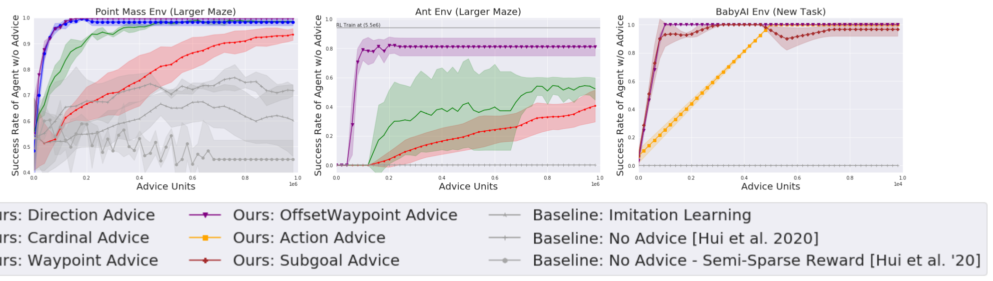

Figure 7: Learning new tasks through distillation. The agent uses an already-grounded advice channel to perform the distillation process from Section 4.2 to train an advice-free agent. Plots show the success rate of the advice-free new agent. We show that distillation is more advice efficient than training with RL, and that more abstract advice forms (waypoints, subgoals, etc.) are more advice-efficient than than lower-level advice.

Finally, we evaluate the ability of grounded advice to provide a more effective medium towards guiding the agent in learning new tasks, as compared to demonstrations or rewards. As we can see in Fig 7, for new tasks in all domains, agents which are trained through distillation from an abstract teacher on average train with 18 times less advice than RL agents while simultaneously achieving an improvement in average asymptotic performance. What is particularly impressive here is that many of the new tasks are more challenging than training tasks (more complex navigation and manipulation problems), as shown from an illustration in the Appendix, and as seen from Fig 7 are hard to learn with RL.

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 keeps getting better. This suggests that higher bandwidth, lower effort communication (like subgoals/waypoints) can often be extremely effective.

6 Discussion

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.

While the current work is a start towards formalizing the broader class of interactive teaching problems, it is restricted to prescriptive coaching. Generalizing this to feedback provided after the fact would be an interesting avenue of future research. Additionally evaluating our approach with human teachers in the loop would test the real world applicability of this work. This would require improvements in efficiency and perhaps the incorporation of more efficient off-policy RL algorithms.

Limitations: This work is limited by the fact has been done in simulation with a scripted teacher rather than real humans. The fact that the agent uses supervised learning to learn behaviors makes it subject to the challenges of supervised learning such as compounding error. Additionally, the advice provided is assumed to be optimal, but going forward a noisy model of human feedback needs to be incorporated.

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. However this work is quite far from real world deployment so we will work towards addressing these challenges before deployment.

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] 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.

[18] 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`.

[19] 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.

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

[21] 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`.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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`.

[28] 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.

[29] 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`.

[30] 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`.

[31] 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`.

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

[33] 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`.

[34] 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`.

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

[36] 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.

[37] 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.

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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]