The Dissertation Committee for Taylor Annette Kessler Faulkner certifies that this is the approved version of the following dissertation:

**Learning Robot Policies from Imperfect Human Teachers**

Committee:

Andrea Thomaz, Supervisor

Manuela Veloso

Peter Stone

Elaine Short

Constantine Caramanis

# Learning Robot Policies from Imperfect Human Teachers

**by**

**Taylor Annette Kessler Faulkner**

**DISSERTATION**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**DOCTOR OF PHILOSOPHY**

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2022

Dedicated to my husband and parents, for their love and support.

# Acknowledgments

I would like to thank Andrea Thomaz, my phenomenal research advisor and leader of the Socially Intelligent Machines (SIM) lab, for always supporting my ideas and having my back throughout the highs and lows of graduate school. Thanks are also owed to Elaine Short, my unofficial secondary advisor, for helping me develop a work-life balance alongside my thesis during her time as a postdoctoral researcher in the SIM lab.

Thank you to the collaborators and labmates that I've worked closely with during my PhD (Guy Hoffman, Mai Lee Chang, Akanksha Saran, Tesca Fitzgerald, Shih-Yun Lo, Adam Allevato, Alex Gutierrez, Priyanka Khante, Yuchen Cui, Ajinkya Jain, Thomas Wei, Kush Desai, Srinjoy Majumdar, Max Svetlik, Kim Baraka, Reuth Mirsky, and others) for creating a supportive and collaborative working environment. A special acknowledgement is also owed to Mai Lee Chang for reading through my thesis, and for her continued support through thick and thin during our last year at UT Austin.

I also have to acknowledge my academic advisors and study group from my undergraduate degree at Denison University, who helped make applying to graduate school a manageable process. Thanks to my undergraduate computer science class team (Trevor Masters and Paige Vosmik) for sticking with me for 3+ years of study groups, class projects, and coding competitions. Thank you

also to Matt Kretchmar and Ashwin Lall, my research advisors during my time as an undergraduate at Denison University who encouraged me to apply to graduate school and provided a wonderful research education that has helped me throughout my PhD program.

Many thanks to my husband Brant Bowers, parents Rob Kessler and Kelly Faulkner, and all of the friends who supported me throughout graduate school: Brant for being the most encouraging and patient partner I could have asked for, my mom and dad for fostering my academic dreams, and the close friends I've managed to keep in touch with throughout graduate school (Samantha Allen, Jess Hoffmann, Natalie Foster, Chris Conard, Greg Ponti, Luc Lisi, Alyss Flynn, Ace Furman, Sara Schroer, Sami Steinkamp, Will Cornell, Alex Farrenkopf, Nic Lyman, Chris Robie, Will Brackenbury, among others) for providing support, distraction, and entertainment when graduate school was too overwhelming. Finally, a thank you to my cats, Lord Albert Stormageddon (Storm) and Smokira Queen of Shadows (Smoke) for emotional support, forced breaks when working from home, and typing assistance during the writing of this document.

# Learning Robot Policies from Imperfect Human Teachers

Publication No. _____

Taylor Annette Kessler Faulkner, Ph.D.
The University of Texas at Austin, 2022

Supervisor: Andrea Thomaz

The ability to adapt and learn can help robots deployed in dynamic and varied environments. While in the wild, the data that robots have access to includes input from their sensors and the humans around them. The ability to utilize human data increases the usable information in the environment. However, human data can be noisy, particularly when acquired from non-experts. Rather than requiring expert teachers for learning robots, which is expensive, my research addresses methods for learning from imperfect human teachers. These methods use Human-in-the-loop Reinforcement Learning, which gives robots a reward function and input from human teachers. This dissertation shows that

**Actively modifying which states receive feedback from imperfect, unmodeled human teachers can improve the speed and dependability of Human-In-the-loop Reinforcement Learning (HRL).**

This body of work addresses a bipartite model of imperfect teachers, in which humans can be inattentive or inaccurate. First, I present two algorithms for learning from inattentive teachers, which take advantage of intermittent attention from humans by adjusting state-action exploration to improve the **learning speed** of a Markovian HRL algorithm and give teachers more free time to complete other tasks. Second, I present two algorithms for learning from inaccurate teachers who give incorrect information to a robot. These algorithms estimate areas of the state space that are likely to receive incorrect feedback from human teachers, and can be used to filter messy, inaccurate data into information that is usable by a robot, performing **dependably** over a wide variety of inputs.

The primary contribution of this dissertation is a set of algorithms that enable learning robots to adapt to imperfect teachers. These algorithms enable robots to learn policies more quickly and dependably than other existing HRL algorithms. My findings in HRL will enhance the ability of robots to learn new tasks from laypeople, requiring less time and knowledge of how to teach a robot than prior work. These advances are a step towards ubiquitous robot deployment in the home, public spaces, and other environments, with less demand for expensive expert data and an easier experience for novice robot users.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

*"Sometimes, I just don't understand human behavior. After all, I'm only trying to do my job."*

—C-3PO (Star Wars: Episode V – The Empire Strikes Back)

Robots that are deployed in human environments, such as homes and public spaces, can benefit from the ability to acquire new skills in order to adapt to their changing surroundings. Robots that can learn from people can potentially adapt and learn more quickly than robots that learn from environmental sensor observations alone. Given that expert time is rare and costly, enabling laypeople to teach robots in the wild will allow more robotic agents to be deployed in human environments. The ability to learn from any person the robot comes across would give robots access to more data and learning opportunities. However, non-experts may be imperfect at supplying learning robots with data. In this dissertation, I present algorithms for learning from a bipartite model of imperfect teachers, in which humans can be *inattentive* and *inaccurate* while giving feedback to a learning robot (Figure 1.1). These algorithms use Human-in-the-loop Reinforcement Learning (HRL) to intake data from human teachers, a method of learning that allows human teachers to give robots feedback or advice on actions rather than providing full demonstrations.

HRL is able to give robots two sources of information: an environmental reward function and additional feedback from human teachers. Robots can use one of these sources to confirm the performance of the other, balancing how much weight a robot puts on each source of information. Many prior methods in HRL work quite well with human feedback but do not fully address human limitations or preferences, assuming that human teachers are constantly available during the learning process, or that the robot has a known prior on how correct the teacher's feedback is [43]. In this body of work, I address these human differences to improve the quality of HRL with imperfect teachers, enabling robots to learn more quickly, with policies that are robust to a wide variety of human input. The two research directives, *Learning from inattentive teachers* and *Learning from inaccurate teachers* are outlined in the following sections. I define *inattentive* teachers and *inaccurate* teachers in Section 3. In all of these works, I first test the algorithm performance in simulation with a simulated teacher in order to measure how the algorithm performs in theory. Then, I test on a physical robot and/or with real people, in order to test how the algorithms perform in practice.

My algorithms are built on an MDP, and the task is thus assumed to be Markovian: that is, the next state only depends on the current state, with no other recorded history factoring in. This means that when teachers come back to the robot to give it attention, they will not need to know a history of actions, but instead can infer the best action for the robot to take based on their observation of the current state alone. However, if people are aware of the

robot's previous actions, they may take this history into account when giving feedback, which is inherently non-Markovian. Additionally, this Markovian assumption may not hold in tasks that are history-dependent, which I address in Chapter 6. Despite these issues, using an MDP will function throughout this thesis as an approximate model of the human-robot interactions that I study.

## 1.1 Learning from inattentive teachers

My work on inattentive teachers includes two algorithms, Attention-Modified Policy Shaping (AMPS) [24] and Active Attention-Modified Policy Shaping (AAMPS) [34]. The AMPS algorithm capitalizes on human attention by increasing exploration when the teacher is available and decreasing exploration while no teacher is present, assuming that teachers are always attentive when present and only require viewing the current state of the robot to give feedback. AMPS allows the robot to learn a policy more quickly and requires less time from teachers than Policy Shaping [30]. In the work following AMPS, AAMPS extends this method to enable the robot to actively request feedback from inattentive teachers. This extension allows the robot to receive feedback at informative points, and takes the burden of deciding when to return to the learning robot off of human teachers. These works improve the **speed** of learning policies using HRL, while requiring less feedback and attention from human teachers. In this work, periods of attention or inattention are predetermined, not sensed by the robot. The assumption of perfect attention

detection allows us to directly compare AMPS with Policy Shaping. However, noisy attention perception, as has been done in prior work [26, 42, 48, 63, 67] could easily be plugged into the AMPS and AAMPS algorithms.

## 1.2 Learning from inaccurate teachers

To learn from inaccurate teachers, I enable robots to decide which teacher-provided information to trust, using additional sources of information such as the reward function in HRL. These works are motivated by circumstances in which people will give consistently bad feedback on some states and consistently good feedback on others. This may happen if teachers are confused about the task goal or how the robot functions. I present an algorithm, Revision Estimation from Partially Incorrect Resources (REPaIR), that can filter incorrect information to usable information for the robot and propose several studies to validate this algorithm. REPaIR saves a maximal cumulative reward value for each state-action pair and uses this memory to compute trust values in human feedback. I also present an algorithm, Classification for Learning Erroneous Assessments using Rewards (CLEAR), that uses a classifier to learn to predict the slope of the learning curve based on observed state-action pairs, to reduce the amount of storage space needed to filter incorrect information. That is, instead of storing reward information for each individual state-action pair, the classifier enables CLEAR to learn to predict task performance based on state features and actions. CLEAR also supplements feedback given by human teachers, in order to more easily extend

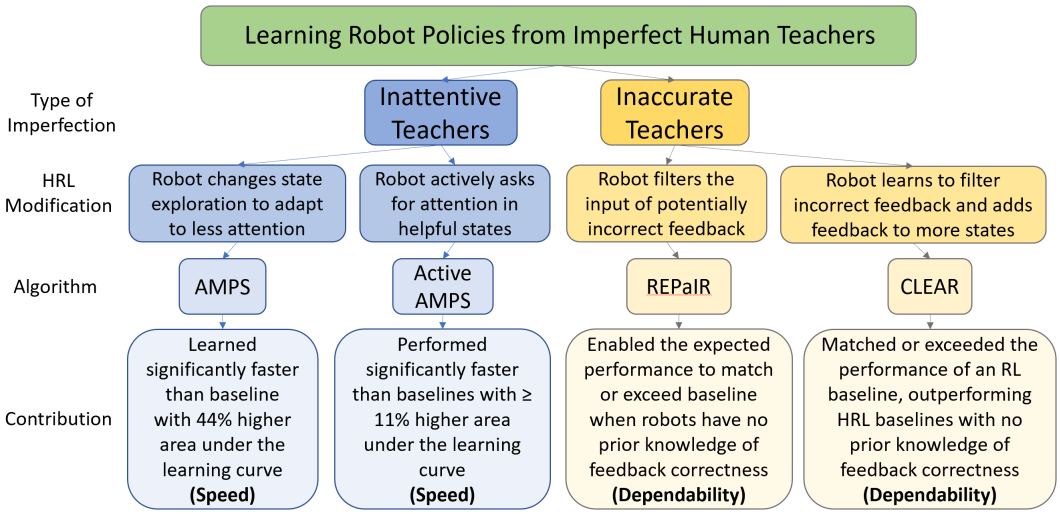



Figure 1.1: My completed body of work, composed of the AMPS, AAMPS, REPaIR, and CLEAR algorithms.

to complex tasks, in which robots may need more feedback than is possible to receive from a single teacher. These methods improve the **dependability** of HRL algorithms, enabling them to perform robustly without prior knowledge or modeling of the correctness of human teachers.

## 1.3 Contributions

Specifically, this dissertation provides the following contributions:

1. A HRL algorithm (AMPS) that changes RL exploration in order to learn significantly faster than a baseline with 44% higher area under the learning curve (**Speed**) [24] (Chapter 4)

2. A HRL algorithm (AAMPS) that enables robots to ask for attention from

inattentive teachers when needed, performing significantly faster than baselines with $\geq 11\%$ higher area under the learning curve (**Speed**) [34] (Chapter 4)

3. A framework for Markov Decision Processes with incorrect feedback [35] (Chapter 3)

4. An algorithm (REPaIR) that filters imperfect feedback to various HRL algorithms, enabling the expected performance to match or exceed baseline when the robot has no prior model of expected human feedback correctness (**Dependability**) [35] (Chapter 5)

5. An algorithm (CLEAR) that filters imperfect feedback to a HRL algorithm in a large state space, as well as adding supplemental feedback, matching or exceeding the performance of an RL baseline, outperforming an HRL baseline when the robot has no prior model of expected human feedback correctness (**Dependability**)

# Chapter 2

# Background and Related Work

*"The need for more research is clearly indicated."*
—Data (Star Trek: The Next Generation, Season 4 Episode 11)

This chapter introduces important background concepts and algorithms and covers relevant prior work. I introduce Reinforcement Learning (Section 2.1.1), the Policy Shaping algorithm [30] (Section 2.1.2), and TAMER+RL [38] (Section 2.1.3). Then, related work in Human-in-the-loop Reinforcement Learning (Section 2.2.1), learning from inattentive teachers(Section 2.2.2), and learning from inaccurate teachers 2.2.3) will be discussed.

## 2.1 Background

The Reinforcement Learning framework is detailed here here, with a review of some relevant algorithms in HRL, which are used as baseline algorithms in several of my works.

### 2.1.1 Reinforcement Learning

Reinforcement Learning (RL) is built on a Markov Decision Process (MDP). MDPs consist of a tuple $(S, A, T, R, \gamma)$:

- $S$: a set of states
- $A$: a set of actions
- $T(s, a, s')$: a transition function giving the probability of transitioning to $s' \in S$ when taking $a \in A$ in $s \in S$
- $R$: a reward function, giving reward $r_{s,a}$ after taking $a \in A$ in $s \in S$
- $\gamma$: a discount factor

The RL algorithm used in the majority of my experiments is Q-Learning, an off-policy method of RL, which uses a learning rate $\alpha$ and a discount factor $\gamma$ to learn Q-values from rewards using a Bellman update over episodes. Often, Q-Learning is used with Boltzmann exploration to encourage the learning agent to explore the environment rather than just exploit the current Q-values, which can lead to the agent getting stuck in local optima [83,84]. Using Boltzmann exploration, the probability of taking any action $a$ in state $s$ is

$$\Pr_q(s, a) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a'} e^{\frac{Q(s,a')}{\tau}}} \tag{2.1}$$

using the learned Q-values $Q(s, a)$ and $\tau$, a temperature parameter.

### 2.1.2 Policy Shaping

Policy Shaping [16, 30] is a form of HRL that enables people to give binary feedback, positive or negative, to a learning robot. I use this algorithm as a baseline throughout my body of work. This feedback shapes the robot's policy, influencing it towards state-action pairs that have received positive feedback from a human teacher. All actions consider the human feedback as the difference in positive and negative values given by the teacher, or $\Delta_{s,a}$. Using $\Delta_{s,a}$ rather than the count of positive feedback on $(s, a)$ compensates for the possibility that teachers may be slightly inconsistent in their feedback on the same state-action pair at different times.

Policy Shaping affects the exploration style of the robot, rather than influencing the rewards or Q-values of the MDP directly. Let the probability of taking any action using exploration methods based purely on the MDP be $\Pr_q(a|s)$. The probability that an action is good using feedback is

$$\Pr_c(a|s) = \frac{C^{\Delta_{s,a}}}{C^{\Delta_{s,a}} + (1-C)^{\Delta_{s,a}}} \tag{2.2}$$

where $C \in [0, 1]$ is a trust parameter, with 0 being complete distrust in the human teacher and 1 being complete trust. When $C = 0.5$, PS reduces to RL with no feedback. In these experiments, I cap $\Delta_{s,a}$ to avoid overflow computation errors in Python 2.7. Using $\Pr_q(a|s)$ and $\Pr_c(a|s)$, the probability of taking any action $a \in A$ in state $s \in S$ while learning is

$$\Pr_p(a|s) = \frac{\Pr_q(a|s)\Pr_c(a|s)}{\sum_{\alpha \in A} \Pr_q(\alpha|s)\Pr_c(\alpha|s)}. \tag{2.3}$$

### 2.1.3 TAMER+RL

TAMER is an algorithm for replacing a reward function with human feedback [36,37]. An extension to TAMER experimented with several methods of combining scalar feedback with a reward function [38]. I consider two such methods that were shown to outperform SARSA [76], an on-policy method of RL. In the following equations, $\hat{H}(s,a)$ is the human's reward function, learned over time using TAMER. TAMER uses a learned predictor of feedback after each state-action pair, in tasks that do not allow immediate feedback from humans after each action [36, 37]. Throughout this dissertation, I will refer to the two versions of TAMER+RL as TAMER-P and TAMER-W, so named for the variables that enable TAMER+RL to weight the importance of $\hat{H}(s,a)$ versus $Q(s,a)$.

1. TAMER-P:

$$P(a = argmax_a[\hat{H}(s,a)]) = p. \tag{2.4}$$

   With probability $1-p$ the original RL agent's action selection mechanism is used. $p$ is gradually diminished over time, so the human's feedback is more influential at the start of the learning process.

2. TAMER-W:

$$\boxed{a = argmax_a[Q(s,a) + w^* \hat{H}(s,a)].} \tag{2.5}$$

## 2.2 Related Work

In this section, I cover background work in Human-in-the-loop Reinforcement Learning (HRL), My work on learning from inattentive teachers relies on changes in attention during human-robot engagement, related to curiosity-driven learning, and actively requesting specific information from teachers, related to active learning. My work on learning from incorrect teachers is related to other research that learns from imperfect information from teachers, and the proposed work is based on adapting to the teaching styles of human teachers.

### 2.2.1 Human-in-the-loop Reinforcement Learning

HRL allows a Markov Decision Process (MDP) to be supplemented with additional input from a human teacher [43]. This input can take many forms, such as binary or scalar-valued feedback [30, 38, 72, 75, 82], advice on future actions [40, 41, 45, 51, 71], or action intervention [68]. Human feedback can also replace the reward function [36]. Other methods of HRL can use full demonstrations [32, 44, 52, 61, 75]. I will cover some of these works in more detail in the following sections.

### 2.2.2 Attention from Human Teachers

Related work in human-robot interaction (HRI) considers human attention (or engagement) to modify robot behavior [48, 62, 88], or attempts to convince people to engage with the robot [14, 73]. However, this work does not modify learning styles, as my algorithms do. Curiosity-driven learning, also known as intrinsic motivation, allows learning agents to explore their environment based on maximizing learning and information potential, not just maximizing rewards or values [2, 17, 56, 70], but these works do not include human teachers, or do not give teachers breaks. Previous work by Oudeyer et al. [56] has combined curiosity-driven learning with human teachers, creating an agent that chooses whether to follow human advice or explore, but this work also assumes that human feedback is always available to the robot, unlike ours. Similar to my algorithms, there has been previous research on active RL [21] without available human teachers, using initial estimates of an MDP to direct exploration. However, this exploration is not based on human feedback. Other prior research on active RL [4, 15, 18–20] queries teachers for feedback in informative states, but assumes that teachers are always present and available to give feedback. There is also work that attempts to ask surrounding people for their attention, although the robot is not trying to learn a task in these cases [53, 65].

### 2.2.3 Learning from Incorrect Information

Prior work has proposed algorithms that compensate for incorrect supplemental human feedback ( [30, 38, 49, 74]) or demonstrations [27, 33, 75, 80] in an RL framework. Other works use static [30] or slowly decreasing reliance on a teacher's feedback over time [38]. There is also research into how to state questions to laypeople in such a way to avoid confusion and incorrect answers about state inference [64]. Sridharan [74] keeps track of policies from a reward function, and one policy from feedback, weighting trust based on the agreement between its policy and the reward function policies. Another method bases the trust in a teacher by comparing teacher advice and currently learned Q-values, as well as the current trust in a deep RL algorithm [45]. These methods may discount good feedback at the beginning of learning, when the Q-values and initial policies are still likely to be incorrect. [49] showed a neural net trained with a loss function that modeled noise as an asymmetric Bernoulli outperformed a neural net trained with binary cross entropy loss in the classification of pixels from aerial images. Their approach relied on *a priori* information on the probability of label flip noise.

There is also research into learning from incorrect information in Learning from Demonstration (LfD) [7]. In Inverse Reinforcement Learning (IRL), the agent is provided with full demonstrations from the teacher and has to estimate the reward function from these demonstrations [13, 22, 31, 91]. However, all of these works require full demonstrations, or additional information such as rankings [13] or the relative frequency of bad demos [31, 91].

## 2.3 Thesis Motivation

Over this survey of prior works, a pattern emerges. While there are state-of-the-art HRL algorithms that perform very well with perfect human teachers, gaps begin to show when people do not behave as ever-present oracles. When teachers are imperfect, robots can behave unpredictably, learn slowly, or learn incorrect behaviors. My algorithms use insights about human behavior to improve the capability of robots learning using HRL, by addressing the cases in which teachers are inattentive or give inaccurate feedback to a learning robot. This thesis focuses on enabling robots to learn with the assumption that humans need breaks and may misunderstand tasks or how the robot functions.

# Chapter 3

# MDP Framework for Inattentive and Inaccurate Teachers

This framework is inspired by Everitt et al. [23], who developed a framework for CRMDPs (Corrupt Reward MDPs), for which the reward signal itself is unreliable. Let $t$ be an individual teacher that gives feedback $F_t(s, a)$ for state $s$ and action $a$, and $N$ be the total number of time steps for a learning algorithm using the IFMDP. Then, potentially imperfect environmental feedback is added to an MDP, creating an Imperfect Feedback MDP (IFMDP), consisting of a tuple $(Att_{t,n}, F_t^*, F_t, S, A, A_{s,seen}, T, R, \gamma)$:

- $Att_{t,n}$: a binary variable in

  $$0, 1$$

  that indicates whether the teacher $t$ is present and watching the robot (1) or not (0) at time step $n \in N$

- $F_t^*(s, a), F_t(s, a)$: *correct* and given teacher feedback/advice functions for teacher $t$, related by $\Gamma$ s.t. $F_t(s, a) = \Gamma(F_t^*(s, a))$. $F_t(s, a) > 0$ indicates positive feedback.

- $S$ : a set of states

- $A$ : a set of actions
- $A_{s,seen}$: a set of actions $a \in A$ from state $s$ that have been observed by teacher $t$ ($Att_{t,n} = 1$)
- $T(s, a, s')$ : probabilities of transitioning to $s' \in S$ when taking $a \in A$ in $s \in S$
- $R_h(s, a)$ : the reward function, as defined by some human $h$
- $\gamma$: a discount factor

$S, A, T$, and $R_h$ are all identical to a standard MDP. In addition, I supplement the MDP tuple with $Att_{t,n}$,$A_{s,seen}$, $F_t^*(s, a)$, and $F_t(s, a)$ to account for inattentive and inaccurate teachers. For inattentive teachers, $Att_{t,n}$ is a binary signal that the robot can reference to determine whether a teacher is paying attention to it. I define attention for this work as a state in which a teacher $t$ is present and watching the robot, assuming that in this case they are paying attention to the robot's actions; this definition does not include a teacher that is present, watching, but not thinking about the robot. $Att_{t,n}$ can be toggled by the teacher $t$, by an experimenter, or by an attention-detection function [26, 42, 48, 63, 67]. An *inattentive* teacher $t$, returning feedback using $F_t(s, a)$, is defined as follows:

$$\boxed{\exists n \in N \text{ s.t. } Att_{t,n} = 0.} \tag{3.1}$$

For inaccurate teachers, I add $F_t^*(s,a)$ and $F_t(s,a)$. There exists some family of reward functions $\mathcal{R}^*$ such that an optimal desired policy $\pi^*$ will be learned. This family of reward functions is defined as $R \in \mathcal{R} \implies R \to \pi^*$. However, $R_h \in \mathcal{R}^*$ is not necessarily guaranteed. $F_t^*$ and $F_t$ are the correct and given teacher feedback functions for some teacher $t$. A corruption function, $\Gamma$, is defined as the difference between $F_t^*$ and $F_t$ s.t. $F_t = \Gamma(F_t^*)$. If $F_t$ is correct, $\Gamma$ will be the identity function.

Here I define *correctness*, a term that is used in this work to describe reward functions and human feedback:

- Correct feedback/advice: the ranking of all values or suggested actions in state $s$ by the teacher is the same as the ranking of all actions in $s$ by the learned value function.
- Correct reward functions: following $R$ produces the policy $\pi^*$.

A correct teacher, as defined above, agrees completely with $\pi^*$. *Correctness* is defined in relation to $V(s,a)$, the state-action values that would be learned from $S, A, T$, and some $R \in \mathcal{R}^*$ if all information was given to the agent. Consider $\pi^*(s) = \text{argmax}_a(V(s,a))$. Thus the desired policy of the teacher must match $\pi^*$ to be correct. I define $F_t*$ as follows:

$$\boxed{F_t^*(s,a) \geq F_t^*(s,a') \iff V(s,a) \geq V(s,a').} \tag{3.2}$$

An incorrect teacher can thus be defined as a function $F_t(s,a)$ where:

$$\boxed{\exists s, a \text{ s.t. } F_t(s,a) \neq F_t^*(s,a).} \tag{3.3}$$

# Chapter 4

# Learning from Inattentive Teachers

*"I want to be alone!"*
—Bender (Futurama: Season 7 Episode 12)

*"Look at me! I want attention."*
—Bender (Futurama: Season 2 Episode 19)

In this chapter, I present my algorithms for learning from *inattentive* teachers. Human-in-the-loop RL methods often assume that the teacher is continuously paying attention, watching and maintaining awareness of a robot's actions. However, the assumption that a human will be constantly available to give feedback is unlikely to hold. These algorithms, Attention-Modified Policy Shaping (AMPS) and Active Attention-Modified Policy Shaping (AAMPS), enable a robot to change its behavior depending on the presence of human attention. With less human attention, robots learn slowly, and may even learn incorrect policies under certain algorithmic conditions. For example, Cederborg et al. showed that, in certain cases, robots should interpret a teacher's lack of feedback as implying approval of actions when using the Policy Shaping algorithm [16]. However, this result no longer holds when teachers are paying intermittent attention, as robots could take bad, even disastrous actions, while

interpreting the inattentive teacher's silence as tacit approval. The algorithms I present in this chapter enable robots to take human attention into account, modifying their policy exploration to control which states are observed by a teacher. This enables the robots to learn faster than baselines, with less human attention.

Continuing learning as usual while no one is present can speed up learning, but can also cause unwanted, unpredictable robot behavior during periods of inattention. In previous approaches to HRL, if no human is available the robot learns from its environment. However, continuing to explore the environment as usual while no human is observing may not be optimal behavior. For example, consider a robot deployed in a home, learning the necessary motions to put away dishes. If the robot has a good model of putting cups away but is still exploring to find more efficient methods, exploring without a person around to observe and potentially stop the robot is likely to result in broken glass all over the kitchen. A better approach might be for the robot to put away cups in a trusted way when left alone, and attempt to learn better actions when supervision is available. [1]

[1]Parts of this chapter have been previously published under *Policy Shaping with Supervisory Attention Driven Exploration [24] ©2018 IEEE* and *Active Attention-Modified Policy Shaping [34] ©2019 International Foundation for Autonomous Agents and Multiagent Systems* .

## 4.1 Attention-Modified Policy Shaping

I present an extension to Policy Shaping, a method of HRL, that takes into account human attention. I define *attention* as the state of a human watching and maintaining awareness of a robot's actions, and consider the ideal case in which the human's attention status is fully observable. An inattentive human is defined as in Section 3. During periods of attention, the robot favors information-gathering actions that allow it to receive feedback about potentially positive states. When unattended, the robot favors actions that have previously received positive feedback during periods of attention. This approach enables the robot to both learn faster in limited-attention scenarios by increasing exploration when supervision is available, and to act more predictable during human inattention by exploiting known "good" actions when in states that humans have previously seen and for which they have provided positive feedback. If there are actions available that a person has approved, the robot will choose from them. I test AMPS in both simulation and on a robot, finding that this method learns faster than Policy Shaping and performs more safely than Policy Shaping while no one is paying attention to the robot.

### 4.1.1 AMPS Methodology: Altering Exploration Based on Attention

I developed an algorithm that changes which actions the robot explores depending on a human supervisor's attentional state. This algorithm combines Q-Learning and Policy Shaping, as described in 2.1.1 and 2.1.2. Both use

**Algorithm 1:** Attention-Modified Policy Shaping

```
while the robot is learning do
    follow Q-Learning
    if person is paying attention then
        with 50% chance, prioritize a ∉ A_seen
        if no available actions not in A_seen then
            follow Policy Shaping
        end
        otherwise prioritize a ∈ A_good
        if no available actions in A_good then
            follow Policy Shaping
        end
    else
        prioritize a ∈ A_good
    end
end
```

Boltzmann exploration [83] with $\tau$ set to 0.5, where $\tau$ is an exploration constant decreased by 1% each learning episode. The Q-learning parameters $\alpha$ and $\gamma$ are set to 0.1 and 0.9 respectively to maximize the performance of Policy Shaping on the chosen task.

AMPS chooses actions based on the teacher's attention, as shown in Algorithm 1. For each state, the agent keeps track of the actions that the teacher has seen, $A_{seen}$ (as defined in Section 3). The agent also keeps track of $A_{good} \subseteq A_{seen}$, the actions that have received more positive than negative feedback. In this work, when a person is paying attention ($Att_{t,n} = 1$ as defined in Section 3), the algorithm randomly chooses with 0.5 probability between taking an action that provides new information (the action is not in $A_{seen}$), and taking an action that might lead to a better part of the state

space (the action is in $A_{good}$). If either $A_{good}$ or $A_{seen}$ is empty when the agent attempts to choose an action from the set, the agent follows the original PS algorithm. When there is no one paying attention ($Att_{t,n} = 0$), the agent maximizes the predictability of its actions by choosing only from $a \in A_{good}$, following the original PS algorithm if no such action is available. When the agent is choosing from a reduced set of possible actions, AMPS calculates the probabilities of each action using Equation 3 with the reduced set rather than all possible actions.

### 4.1.2 AMPS Simulation Experiment: Testing response to attention

I compare AMPS with Policy Shaping on a simulated cup placement task. The robot's goal is to push a cup to a desired location on a table, without pushing the cup off the table. This task could be used to put away cups on a shelf in specific locations; cups on the edge of a shelf are easier for humans to reach at a later point. The table is represented by a 6 by 8 grid in simulation.

The goal location for the cup, $loc_G$, is on the edge of the table, at grid square (5,3) with the grid indexed from zero. This task is well-suited to PS because without human feedback, RL will avoid the edges of the table during learning since they are near dangerous states. PS allows people to guide the robot towards $loc_G$ to allow faster learning. The agent learns the cup placement task using AMPS and PS. The start location of the cup (2,1) and $loc_G$ remain the same throughout.

The problem MDP is formulated as follows:

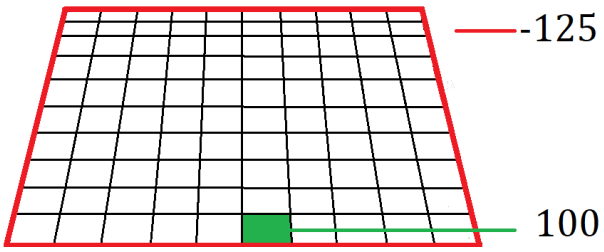



Figure 4.1: Example task environment for AMPS

- $S = (x, y)$, the location on table grid.
- $A = \{\text{north, south, east, west, end}\}$, where the first four actions represent a push in that direction and "end" finalizes the position of the current cup and generates a new cup on the table.
- $T =$ each action pushes one grid square in the specified direction.

The reward is +100 for ending on $loc_G$, where this reward is given as the robot pushes the cup onto the location and taken away if it is pushed off of the location. There is a penalty of -125 if the cup falls off the table. All other states have a penalty of -1 to encourage quick travel to the goal.

An oracle represents a human teacher, giving positive feedback when the agent moves towards $loc_g$ and negative feedback when the agent moves away from $loc_g$. The oracle has two modes: "attentive" and "inattentive". The "inattentive" oracle never gives feedback, while the "attentive" oracle gives feedback 90% of the time, comparable to a human teacher who may not provide complete feedback even when paying attention.

### 4.1.3 AMPS Simulation Results: AMPS learns faster than baseline

Figure 4.2 shows the learning curves for AMPS and PS with the oracle paying attention for two sessions of ten episodes. The shaded sections of the background indicate attention from the oracle. AMPS performs comparably to PS during the first round of attention, but strongly outperforms the prior approach during the period of inattention that follows. In subsequent episodes without attention, performance is greatly improved. The average area under the AMPS reward curve (Mean ($M$) = 7024.025, Standard Deviation ($SD$) = 548.566) is 44% greater than the average area under the PS reward curve ($M = 4877.61, SD = 1357.4$), $t(198) = 14.587, p < 0.05$ (using Welch's t-test). These results suggest that AMPS is learning good actions to take during attention by exploring the environment and exploiting the oracle's feedback, allowing the performance while unattended to be more predictable.

Figure 4.3 shows the result of adding more attention from the oracle throughout the learning process. The difference between the AMPS and PS learning curves decreases as more attention is added, as PS is able to learn more quickly by receiving more feedback. When the oracle pays attention 50% of the time, the percent increase between the average area of PS ($M = 5441.34, SD = 775.347$) and AMPS ($M = 6853.575, SD = 679.433$) is 25.95%, $t(198) = 13.63, p < 0.05$ (using Welch's t-test). When the oracle pays attention for the entire learning process, the percent increase between the area of PS ($M = 6445.975, SD = 546.658$) and AMPS ($M = 6832.095, SD = 404.731$) is 5.99%, $t(198) = 5.648$, $p < 0.05$. With more aggressive exploration, PS

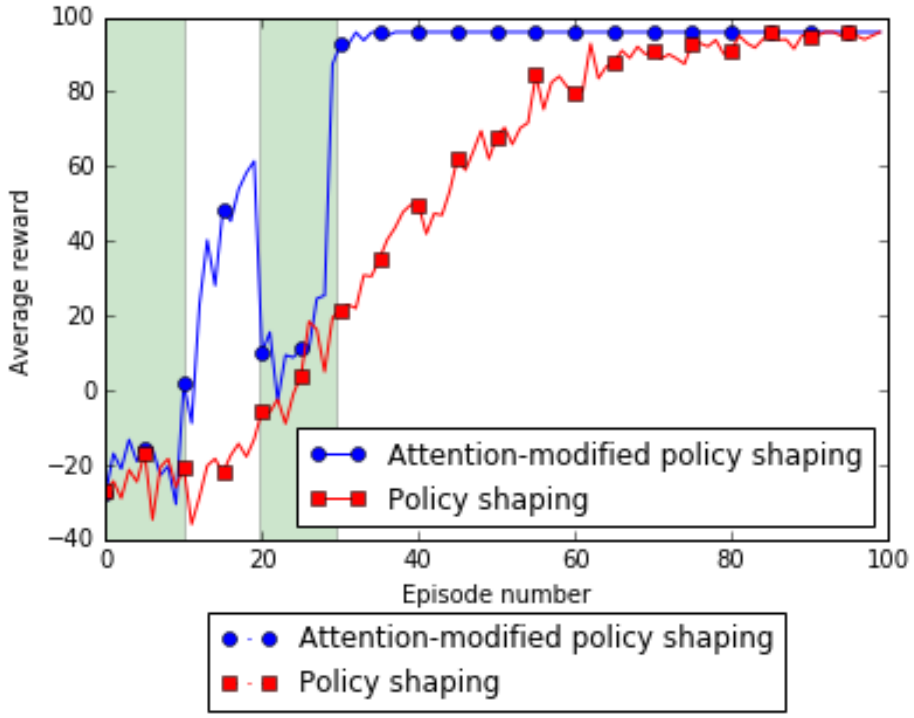



Figure 4.2: Total rewards during learning for 100 episodes. All rewards are averaged over 100 runs. The shaded background indicates attention.

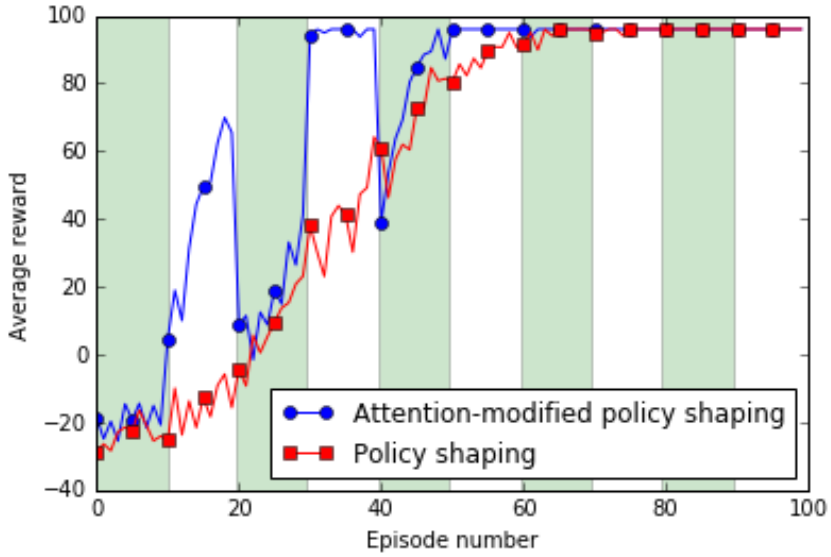
could potentially achieve the same average rewards as AMPS during constant attention. However, in addition to faster learning under intermittent attention, the benefit of AMPS is that while this method explores during periods of attention, it falls back to exploitation of human feedback while no one is paying attention, which enables more predictable performance while unobserved.

### 4.1.4 AMPS Robot Experiment: Testing learning response to human attention with robot study

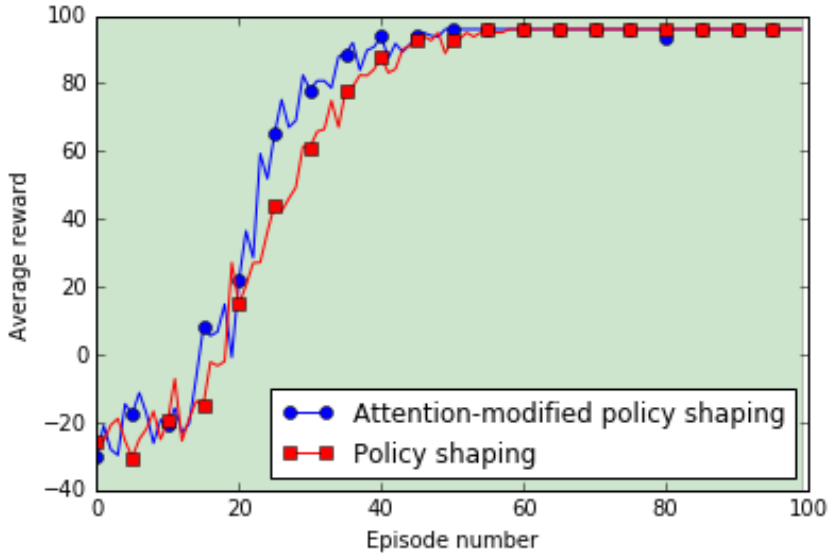
I also tested AMPS with naive users supervising a robot performing the cup-pushing task in the real world. The robot pushed a cup on a table divided into a 6 by 8 grid, with a goal location on the edge of the table. Based on the simulation results, I hypothesized that AMPS would achieve higher rewards during periods of inattention and a greater total reward over all episodes than PS.

This task was implemented on a mobile manipulator robot with a Kinova JACO arm with 7 degrees of freedom and a Robotiq 2-finger adaptive gripper, shown in Figure 4.4. To push the cup, the robot placed its closed gripper inside the cup and moved it a predetermined distance forward, backward, right, or left. The state of the cup was calculated by the position of the gripper over the table by determining in which grid square the robot's gripper location falls. The table was always placed in the same location in front of the robot.

The robot stated the direction in which it planned to move the cup




(a) Oracle pays attention 50% of the time.




(b) Oracle pays attention throughout learning process.

Figure 4.3: Total rewards during learning for 100 episodes with varied attention. All rewards are averaged over 100 runs. The shaded background indicates attention.

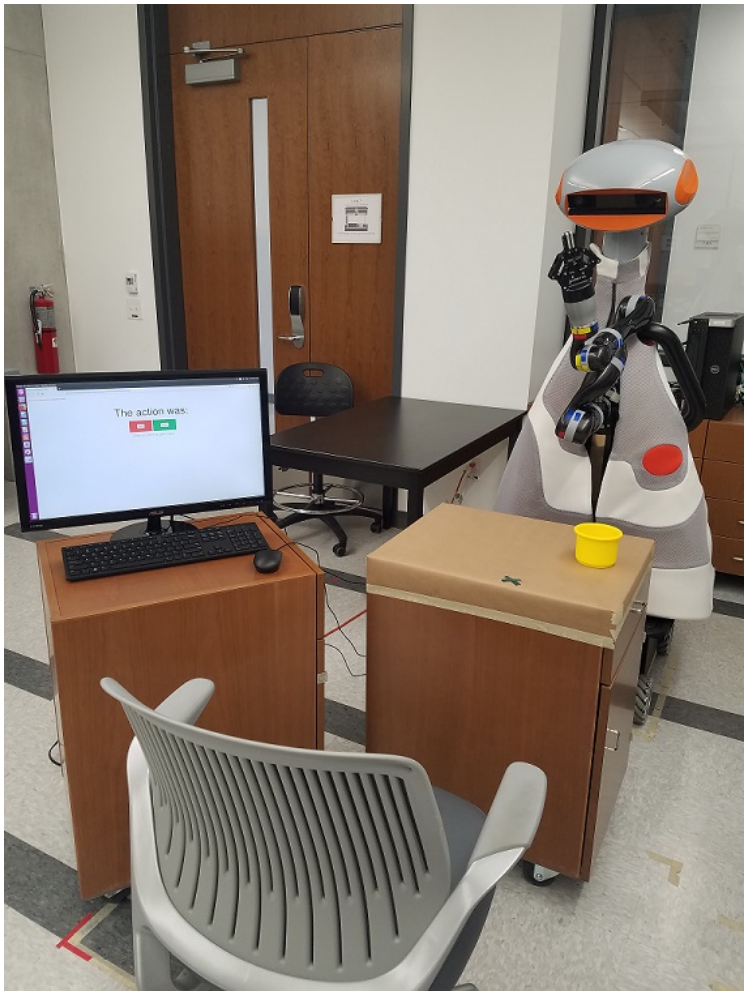



Figure 4.4: Robot used during AMPS experiments.

before attempting the move. During the task, if the robot tried to push the cup in a direction but failed due to the cup catching on the table or a manipulator malfunction, or the cup fell off the table, the experimenter moved the cup to where the robot expected it to be given the robot's statement. If the robot arm caused an error that stopped the learning process, the experimenter restarted learning from the last saved episode. This only happened once during the experiments, on a round of inattention. To control the length of the study, I capped the number of moves per learning episode to twenty pushes. If twenty pushes were reached, the robot asked for the cup to be placed back at the start position.

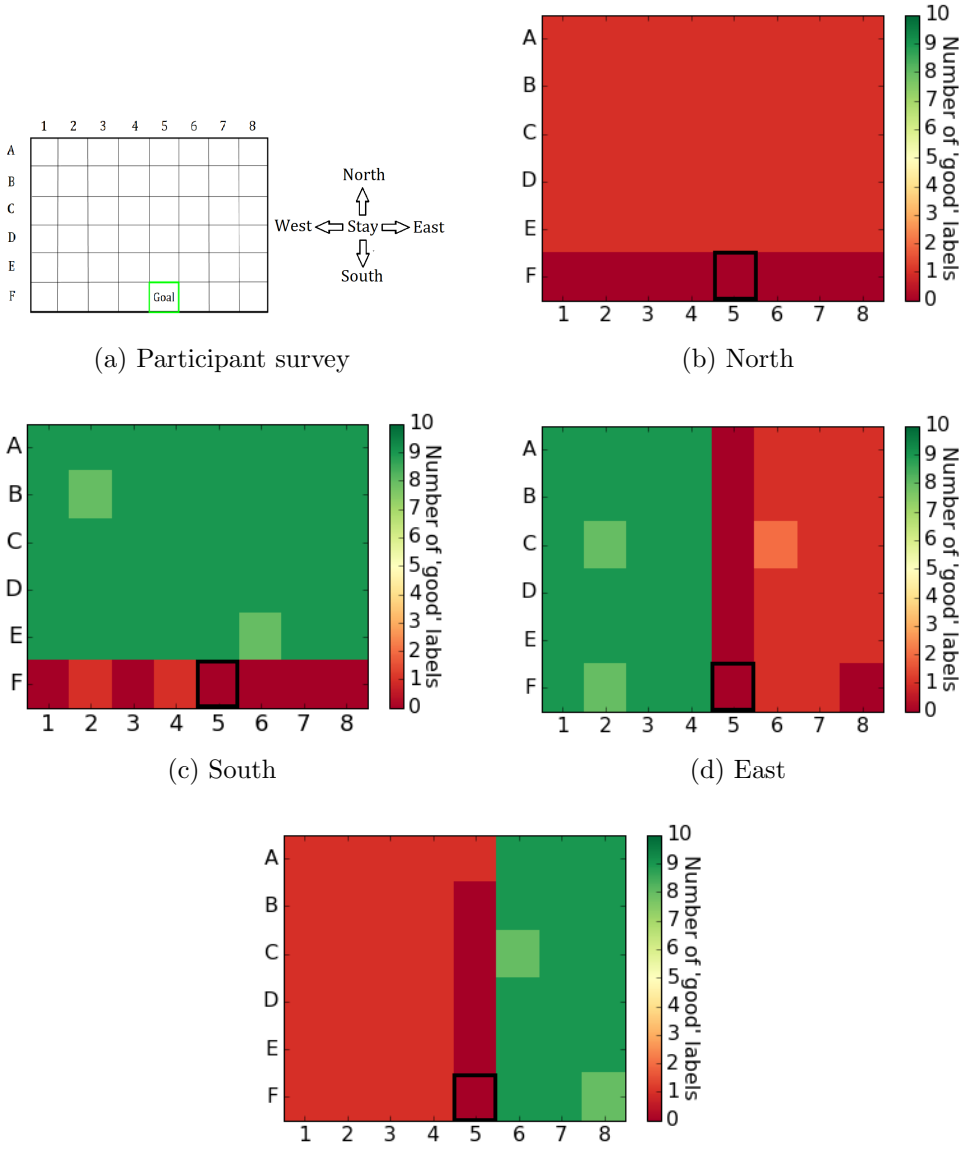
The goal and start locations for the cup were marked on a tabletop. An computer interface was provided with a "Bad" and a "Good" button that could be clicked to send positive or negative feedback to the robot. After taking an action, the robot waited for a response and assumed that no response is given after a timeout. I brought in participants from the campus community to observe the robot and provide feedback while the robot learned the cup pushing task. Each participant observed either the AMPS or PS algorithm. I asked people to click the "Bad" button if they thought an action was bad and the "Good" button if they thought an action was good, paying attention only to the direction of the most recent push action.

Participants were instructed to give feedback for the first ten episodes, ignore the robot for five episodes, come back to give feedback for another four episodes, and then let the robot learn on its own for one more episode. During

the periods of inattention, participants were asked to sit behind a curtain out of view of the robot, and complete a survey designed to capture how they were making decisions about feedback. Each participant looked at an image of a grid with a goal state highlighted in green (Figure 4.5a). For all 48 grid squares in randomized order, they were asked to say whether each action choice (north, east, south, west, and stay) from that square was a "good," "bad," or "neutral" action. After four participants, two of which were used in the data analysis, I noted that there was occasional directional confusion, so the instructions were clarified by explicitly listing the grid square the cup would be in before and after the action. Figure 4.5 b-e shows a heat map of the participants' responses, where red indicates a low number of "good" markings and green indicates a high number of "good" markings.

### 4.1.5 AMPS Robot Results: Robot performs faster with AMPS given time

Figure 4.6b shows the rewards for each episode over all participants. Fourteen participants came in for the study, and four participants were dropped due to robot or human error. Figure 4.6a shows the average rewards for each episode over all participants. To find the average rewards for episodes twenty-one through one hundred and fifty, the robot's learning progress was saved after each participant leaves, and then learning was finished in simulation using the previously described simulation environment. The simulation ran one hundred times for each user, which gives the average performance of both AMPS and PS over multiple trials. Simulating this process multiple




(a) Participant survey

(b) North

(c) South

(d) East

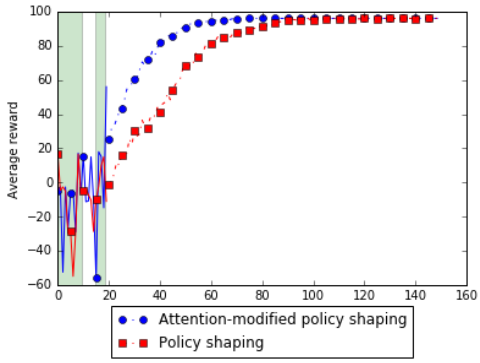
(e) West

Figure 4.5: Participant survey feedback for robot actions, with goal circled in black.

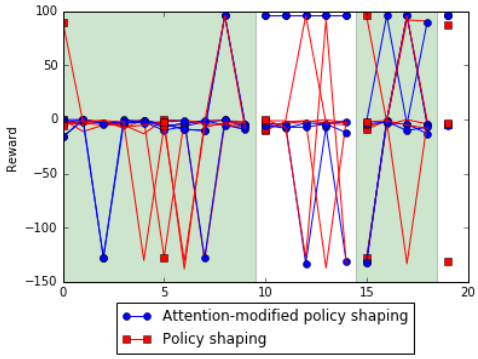
times allows us to show how AMPS will be expected to perform on average.

The average area under the AMPS learning curve during the time that the participant was in the lab (the first twenty episodes) ($M = -127.4, SD = 348.566$) is slightly higher than the average area under the PS curve during the first twenty episodes. ($M = -180.9, SD = 141.216$), $t(8) = 0.285$, $p = 0.787$. The average area under the AMPS simulated learning curve from episodes 20-150 ($M = 11491.252, SD = 818.651$) is higher than the average area under the PS simulated learning curve ($M = 10103.123, SD = 1130.163$), $t(8) = 1.989$, $p = 0.085$. Figure 4.6b shows that there is significant noise in the learning progress of the agent during the first twenty episodes, caused by random factors in RL that cause variation in the rewards received early in the learning process. However, an improvement can still be seen during the second period of inattention. The area under the simulated AMPS learning curve also has a lower variance than that of the simulated PS learning curve.

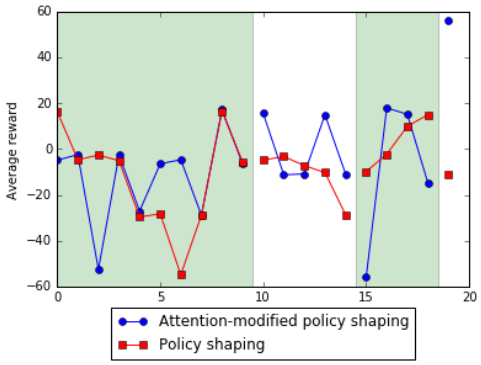
Figure 4.7 shows that algorithm performance for both AMPS and PS varies with amount of feedback given per user. The amount of feedback ranges from 47 to 88. The participants' feedback to the robot during the experiments closely matched the feedback of the oracle used in simulation, in which feedback was positive if the cup moved towards the goal location and negative if it moved away from the goal location. The survey responses suggest that the simulation results are indicative of the performance of the simulator with a human oracle (see Figure 4.5). Two participants did not give feedback for state F8.




(a) Results averaged over all participants. The dashed line represents simulated results. The first twenty episodes are completed during the human study.




(b) The first twenty episodes of Figure 4.6a, with each participant's data shown over 20 episodes.




(c) The first twenty episodes of Figure 4.6a, with average values shown.

Figure 4.6: Total rewards during learning for 150 episodes. The shaded background indicates attention.

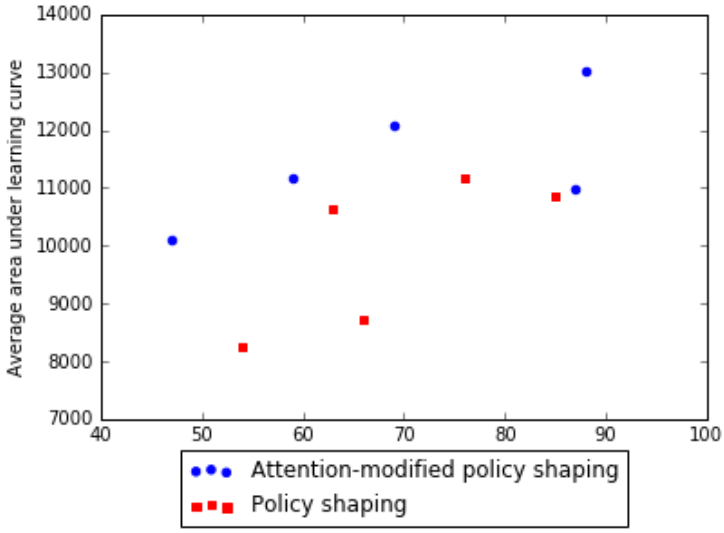



Figure 4.7: Amount of feedback given by participants and the resulting areas under the learning curves

## 4.2 Active Attention-Modified Policy Shaping

While AMPS allows teachers to take breaks from teaching, the burden is placed on the user to decide when to pay attention to the robot. This burden may cause the teacher may be distracted from their other task. Furthermore, if they are unable to check in on the robot they may miss important moments in the learning process, during which feedback would have been useful. Especially with naive teachers, the selected actions may not be the most beneficial for the learning process.

Consider a person cleaning dishes and teaching a robot to put away plates and cups. The person begins by giving the robot feedback while it puts away four cups in a row. The person then goes to the sink with their back

turned to wash dishes. While doing so, they miss the robot attempting to put away a plate for the first time. If the robot had actively decided to ask the teacher for attention during this attempt, the teacher's time could have been balanced better towards giving useful feedback and washing dishes. However, allowing the robot to interrupt the teacher arbitrarily could become disruptive and prevent the teacher from accomplishing other tasks. Therefore, an algorithm that chooses informative times to interrupt the teacher is desirable.

To alleviate the decision-making burden from human teachers, Active Attention-Modified Policy Shaping (AAMPS) actively asks for attention from a teacher in low-information areas of the state space, when there is uncertainty about the teacher's feedback. This modification enables robots to learn even faster than AMPS does, while using less feedback and taking less human attention. Using AAMPS, a robot asks for attention for states in which it is uncertain of the teacher's feedback, with spaces of at least length $t$ in between each request for attention. First the robot checks how long of a break the teacher has had. If it is long enough, the robot checks its certainty of the feedback the teacher might give in the next action. If it is uncertain, it will request attention and feedback. This method removes the responsibility of deciding when to provide feedback from the teacher, enabling the robot to learn quickly while allowing the teacher to spend time on other tasks.

I tested AAMPS both in simulation and in a human study with a robot, comparing to AMPS and other algorithms, finding that AAMPS learns a desired policy more quickly than AMPS in simulation, with an increase of 11.0%

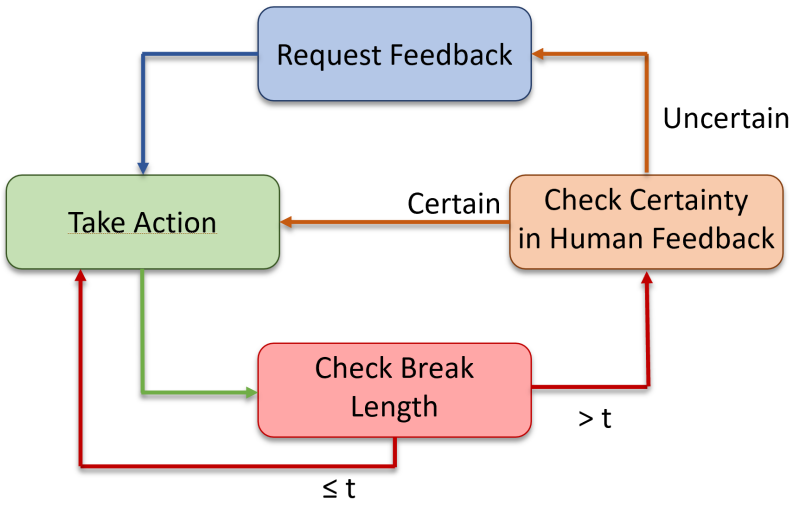



Figure 4.8: AAMPS pipeline, showing steps after taking each action

in area under the learning curve. Furthermore, AAMPS requires attention in 89.9% fewer states than AMPS, as attention is only received in states which require more information. In the human study, I find that AAMPS allows people to complete 77.52% more work on a secondary task than AMPS while the robot receives 48.54% less feedback.

### 4.2.1 AAMPS Methodology: Enabling robots to request attention

This work uses Policy Shaping [16, 30] as a baseline method, described in more detail in Section 2.1.2. For this work, $\tau$, the temperature parameter for Boltzmann exploration is set to 0.3. $C$, the trust parameter for Policy Shaping, is set to 0.9, meaning that we trust the teacher to give correct feedback 90% of the time for the task.

The attention-requesting problem is formulated in the following way.

The robot requests feedback when it is unsure of any positive actions to take in a state, and spaces the requests for attention in order to allow breaks from teaching. This algorithm is shown in Algorithm 2.

In order to define when the robot is unsure of a positive action to take, $\Delta_{s,a}$ is used as in the Policy Shaping algorithm: the difference between positive and negative feedback on state $s$ and action $a$. A confidence threshold $\delta$ is set such that when $\Delta_{s,a} \geq \delta$ for any $a \in A$, the robot considers $(s, a)$ a good state-action pair, as $(s, a)$ has received $\delta$ more positive feedback than negative feedback. When $\Delta_{s,a} \geq \delta$ for any $a \in A$, the robot proceeds to learn without asking for attention, as it is confident that it knows at least one action that the teacher has approved in state $s$. For this work, $\delta = 1$, so that as long as one action has received more positive than negative feedback in state $s$, the robot will no longer ask for attention in state $s$.

If $\Delta_{s,a} < \delta \; \forall \; a \in A$, the robot can ask for attention. In this case, no action has received more positive than negative feedback in state $s$. Therefore, the robot does not know any actions to take that have been approved by the teacher. After attention has been requested in such a state, I assume in this work that the robot receives attention from the teacher. During attention ($Att_{t,n} = 1$), like AMPS, the robot attempts to take actions in $A_{unseen}$ or $A_{good}$ with equal probability.

There is also a time threshold $t$, which limits how often the robot can ask for the human teacher's attention. After each request for attention, the robot must wait for at least $t$ actions before asking for attention again. This

time threshold allows teachers to take predetermined breaks from teaching the robot, so they do not have the robot asking for feedback and interrupting them too often. In this work, $t$ constant action count in order to space attention requests evenly over the length of time that the robot learns. In future work, the variable $t$ could also be non-constant. For example, $t$ could increase over time in order to concentrate feedback at the beginning of the learning curve.

### 4.2.2 AAMPS Simulation Experiment: Testing performance with attention requests

In simulation, I compared AAMPS to several baselines: Q-learning, Policy Shaping, and a simulated variant of AMPS denoted "AMPS Interval". AMPS Interval is equivalent to AMPS with a simulated teacher giving feedback every $t$ rounds. This is more frequent feedback than a person would likely give over 100 episodes of learning. I hypothesized that because AAMPS chooses informative states for feedback, the robot will learn more quickly per unit of feedback. $C$, the Policy Shaping parameter indicating trust in the received feedback, is 0.9 in all experiments. $C$ is held constant across all algorithms, so even if feedback is accurate more or less than 90% of the time, the setting of $C$ does not affect algorithm comparison.

The robot learned a sorting task with four cups, half one color ($k_1$) and half another ($k_2$), in which the robot must sort the cups by color into boxes $b_1$ and $b_2$, in which $k_1$ goes in $b_1$ and $k_2$ goes in $b_2$. The state set $S$ consists of all possible placements of the cups in and out of boxes. The set $A$ of the robot's

**Algorithm 2:** AAMPS

```
S, A = states,actions;
A_unseen = unseen state-action pairs;
A_good = state-action pairs with positive feedback;
t = time threshold;
while learning do
    s = current state;
    if time since last attention request > t then
        if ∃a′ ∈ A s.t. Δ_{s,a′} ≥ δ then
            request_attention();
            p = random var;
            if p ¡ 0.5 then
                action_choices = all a_i ∈ A_unseen;
            else
                action_choices = all a_i ∈ A_good;
            end
            if action_choices = ∅ then
                action_choices = all a_i;
            end
            a = choose Policy Shaping action from action_choices;
        end
    else
        action_choices = all a_i ∈ A_good;
        if action_choices = ∅ then
            action_choices = all a_i;
        end
        a = choose Policy Shaping action from action_choices;
    end
    take_action(a);
    f = get_feedback();
    update_policy_shaping(f);
end
```

action choices includes:

- "Place": place a cup (color $k_1$ or $k_2$) in box $b_1$ or $b_2$
- "Remove": remove a cup (color $k_1$ or $k_2$) from box $b_1$ or $b_2$
- "Restart": pronounce cups sorted and restart the task (can be done at any stage of sorting)

Each episode of the task is only finished when the robot chooses the action "restart", not when the blocks are physically sorted. Therefore the robot learns to sort and then restart. Small negative rewards of -1 are given at each action to encourage reaching the goal state quickly. A reward of 100 is given when the blocks are sorted correctly *and* the robot chooses to "restart". If the robot chooses to restart but the blocks are not correctly sorted, a reward of -10 is given to discourage incorrect restarts.

I created an oracle to use instead of human feedback in simulation. This oracle gives feedback as follows:

- Positive: if placing cup of color $k_1$ in $b_1$ or of color $k_2$ in $b_2$
- Positive: if removing cup of color $k_1$ from $b_2$ or of color $k_2$ from $b_1$
- Positive: if restarting and blocks are correctly sorted
- Negative: otherwise

I compared AAMPS to AMPS Interval, Q-learning, and Policy Shaping. The parameter $t = 2$ for AAMPS and AMPS Interval, so that the robot can at most ask for attention in one out of 3 states. Policy shaping receives attention once every three actions to fairly compare to AAMPS and AMPS Interval. Q-learning does not receive feedback.

For each algorithm, the area under the learning curve is calculated. Each learning episode ends when the robot chooses the "restart" action. The highest reward the robot can receive in a single episode is 96 when the robot places all four cups correctly, receiving a reward of -1 each time, then chooses to restart the task, receiving a reward of 100. The lowest reward the robot could possibly receive in a single episode is negative infinity, as it could take any number of bad actions and then choose to restart, receiving a reward of -10. I hypothesize that AAMPS will learn more quickly than AMPS and AMPS Interval, as it chooses more informative actions for feedback. Each algorithm learns for 100 episodes of the task. These results are averaged over 1000 trials for 100 task episodes each to smooth out random variations in learning speed.

### 4.2.3 AAMPS Simulation Results: AAMPS learns more quickly and with less feedback than baselines

I compared AAMPS, AMPS Interval, Policy Shaping, and Q-learning. The resulting graph of rewards received per task episode is shown in Figure 4.9. The area under each curve (AUC) of total reward over episodes 0-99 was calculated using the composite trapezoidal rule. I found that AAMPS

received more reward on average than AMPS Interval, Policy Shaping, and Q-learning. AAMPS had an average area of 8880.6 under the learning curve, AMPS Interval had an average area of 7999.7, Policy Shaping had an average area of 6987.1, and Q-learning had an average area of 5738.0. Thus AAMPS had an increase in area under the learning curve of 11.0% compared to AMPS Interval, 27.1% compared to Policy Shaping, and 54.8% compared to Q-learning. I found the differences between these algorithms to be statistically significant ($F(3, 3996) = 10641.7, p < 0.0001$) using a one-way ANOVA. Post-hoc tests using Welch's t-test show statistically significant differences between AAMPS and AMPS Interval ($t(1917.7) = 65.4, p < 0.0001$), AAMPS and Policy Shaping ($t(1544.5) = 108.2, p < 0.0001$), and AAMPS and Q-learning ($t(1480.8) = 169.2, p < 0.0001$).

I also compared the amount of feedback each algorithm received on average. While each algorithm learned for exactly 100 episodes, the number of total actions per episode varies. AAMPS received attention on 18.7 actions on average, AMPS Interval received attention on 184.7 actions on average, and Policy Shaping received attention on 203.1 actions on average. Q-learning received no feedback. Policy Shaping receiving more attention implies that even though all algorithms learned for the same number of episodes, Policy Shaping took more actions overall than AMPS Interval.

AAMPS had an decrease in feedback of 89.9% compared to AMPS Interval and 90.8% compared to Policy Shaping. I found the differences between these algorithms to be statistically significant (F(2,2997)=117794.3, $p$¡0.0001)

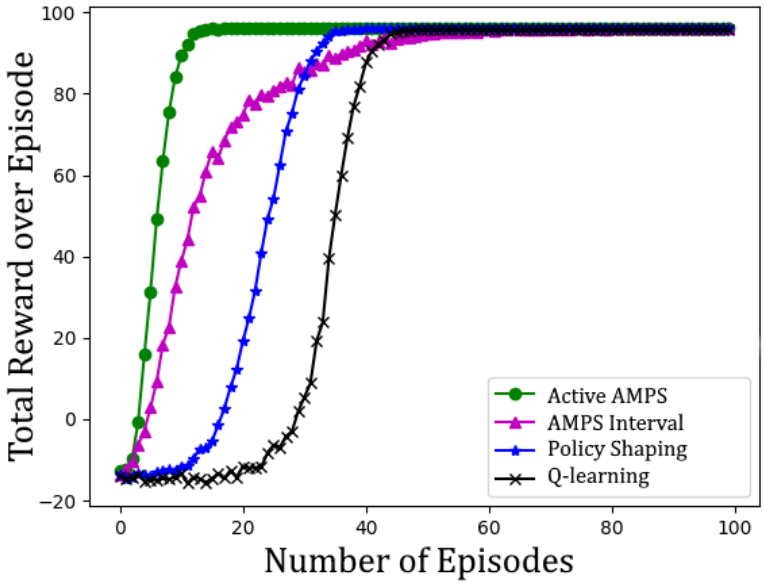



Figure 4.9: Simulated algorithm comparison of rewards gathered over each episode. All algorithms were run for 100 episodes.

using a one-way ANOVA. Post-hoc tests using Welch's t-test show statistically significant differences between AAMPS and AMPS Interval ($t(1982.1) = -595.2, p < 0.0001$) and AAMPS and Policy Shaping ($t(1433.8) = -386.8, p < 0.0001$).

### 4.2.4 AAMPS Robot Experiment: Testing attention requests and human response

I ran a within-subject human study on twelve participants with a physical robot to test human aspects of AAMPS. Three algorithms were tested: AAMPS, AMPS, and AMPS Interval. As in the simulation experiments, AMPS Interval receives attention from the participants after every $t$ rounds. The robot learned for twelve actions in each algorithm, beginning with the

| **State** | [ ], [ ] | [ ], $[k_2]$ | $[k_1, k_2]$, [ ] | $[k_1]$ , $[k_2, k_2]$ | $[k_1, k_1]$, $[k_2]$ | $[k_1, k_1]$, $[k_2, k_2]$ |
|---|---|---|---|---|---|---|
| **Action** | $k_2$ in $b_2$ | $k_1$ in $b_1$ | $k_1$ in $b_1$ | $k_1$ in $b_1$ | $k_2$ in $b_2$ | restart |

Table 4.1: Pretrained state-action pairs in $A_{good}$. The first set of state brackets represents box $b_1$, and the second represents box $b_2$.

same Q values, $A_{good}$, and $A_{seen}$ each time. Fifteen total state-action pairs were in $A_{seen}$, with six in $A_{good}$. The state-action pairs in $A_{good}$ are shown in Table 4.1. Note that each algorithm learns for twelve *actions* in the human study, not *episodes*. All simulation learning was done over 100 full episodes, for which each episode is multiple (potentially over twelve) actions.

The robot completed the same sorting task as it did in simulation. The cups were kept in a holding area at the back of the table and placed on one side or another for sorting. These sides were labeled with the correct color. Pre-recorded actions using kinesthetic teaching were used to pick up and place cups. If the robot failed to pick up a cup during the study but continued the placing or removing motion, its gripper still pointed to the goal location of the cup at the end of the motion. The researchers placed the cup in the goal location in these scenarios, telling the participants to judge the action as if the robot had placed the cup there. Each action took slightly over a minute depending on the position in which the cup was placed.

The real-world "restart" action was executed by the the experimenter after the robot stopped moving, signaling that it thought the cups were sorted. This decision was made to allow sufficient rounds of learning to occur within the one hour long study. The restart action took a variable amount of time, up

to about one minute and eighteen seconds. After each action that the robot took while a participant was watching, it said "Done with action" to signify that it was waiting for feedback.

This study was run with twelve participants from ages 18-30. Five participants identified as female, six as male, and one as agender. The three algorithms (AAMPS, AMPS Interval, and AMPS) were counterbalanced over participants, with the list of all six possible orderings randomized in order over every six participants, so all six orderings were completed after each six participants. The robot and study setup are shown in Figure 4.10.

First, each participant gave informed consent. Then, they were told to balance their time between a distractor task of copying a list of words by hand and teaching the robot to sort the cups. They were given instructions on using the feedback system, giving positive or negative feedback to the last action the robot took, and told that we would count the words they were able to copy. Each participant was told that they would complete three rounds of trying to complete both tasks, and would be given different instructions before each round. Each round corresponded to a different algorithm that the robot was running. After each round, each participant completed a short survey. I describe these steps in greater detail below.

Each algorithm was pretrained using AMPS Interval for 47 actions, which asks for feedback every $t$ actions. This was done to bring AAMPS to a point where the number of feedback asked for per round diverged from AMPS Interval, as eventually AAMPS asks for less and less feedback until it stops

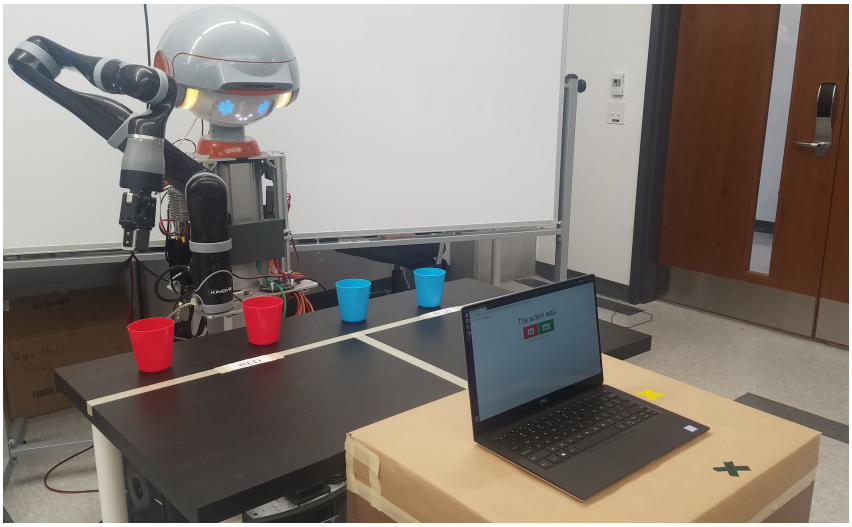


Figure 4.10: Robot performing sorting task used in human study

as described in Section 4.2. In Figure 4.11, we see the number of times the robot asks for attention using AAMPS versus AMPS Interval on one run-through of learning for twenty episodes, which was used for pretraining the algorithms for the human study. AAMPS started asking for attention less often at action 21, when AMPS Interval asked for attention and AAMPS waited, as it had received positive feedback for its current state-action pair. AAMPS fully stopped asking for attention from the teacher at action 73. I used the Q-values, $A_{seen}$, and $A_{good}$ learned after action 47 of AMPS Interval, as this is halfway through the divergence period of the attention requests. Pretraining minimizes the burden to participants by reducing the time needed from them during the study, and focuses the study on the part of the learning process where the frequency of question-asking in AAMPS drops.

The participants divided their time between a distractor task at one

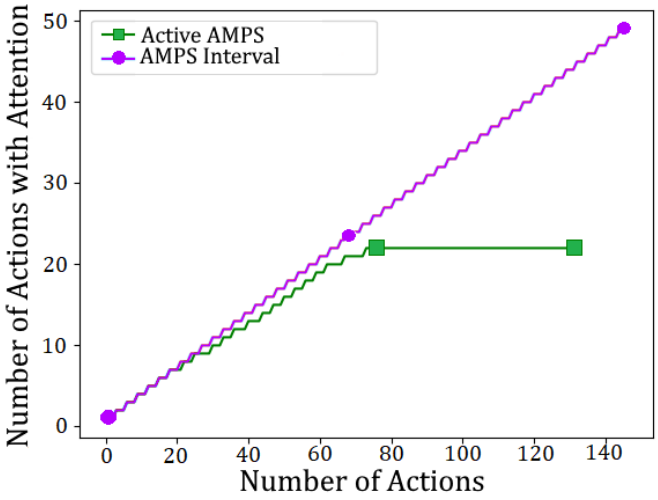



Figure 4.11: Divergence of attention requests between AAMPS and AMPS Interval for one learning run. Requests for attention diminish over time using AAMPS.

work station and teaching the robot at another. The distractor task was structured so that we can measure how much of the task is completed, with participants copying pages of eight-letter words taken from the NLTK corpus [11] printed in order, using pages "airproof-anophyte", "outmount-oxidator", and "labordom-lionizer". These pages were given in the same order to each participant, so that "airproof-anophyte" was the sheet of words to copy for the first algorithm, and so on.

At the start of each "round" (start of new learning algorithm), a new word sheet and blank copying paper were given. Participants were told that each round would take approximately 15 minutes to complete and to balance the two tasks of teaching the robot and writing down words. Participants were

told that their progress teaching the robot and copying words in one round would not carry over to any future rounds. The word copying work states was faced away from the robot, but participants were allowed to glance over at the robot while copying words. In order to not attract undue attention from participants while they were completing the distractor task, the robot did not speak during these times. However, the sound of cups being placed on the table could be heard from the word copying work station.

Since the robot was beginning with the same "sorting knowledge" (Q-values) at the beginning of each algorithm, the participants were told that the robot would be learning to sort differently colored cups each round (red-green, green-blue, red-blue) to visually show the robot starting over at each round. The participants were instructed that the goal of the robot was have all four cups sorted and have the robot say "I believe the cups are sorted. My action is leaving the cups here." Participants gave feedback by clicking a green and red buttons with "Good" and "Bad" text respectively on a computer screen. During AMPS, since the participant could give feedback to the robot for multiple actions in a row, the robot asked after each action if the participant would like to stay and watch another action. If so, the participant clicked a button saying they wanted to stay. Otherwise, they clicked a button saying they wanted to leave. This button enabled the robot to continue learning without checking for attention on the next action, giving the participants time to leave the teaching station.

After allowing participants to practice giving feedback on a few cup

sorting scenarios, the experimenter gave the following instructions before each algorithm, asking the participants to stand in between the robot and the word copying task area until the round began in order to avoid biasing them towards starting at the robot or word copying station.

> **AMPS**: For this round, you can choose when to watch the robot and give feedback, and when to copy words. Feel free to spend as long at each task as you feel fit. You can switch tasks as often as you would like. After each action that you watch, you can press the button to say that you would like to stay and watch another action, or you would like to leave.

> **AMPS Interval and AAMPS**: For this round, the robot will tell you when to give feedback by saying "Please give me feedback," and when to go back to the word copying task by saying "I will learn on my own now." Please listen to the robot's instructions.

The robot detected attention from participants by checking if they were standing in front of the sorting table. Participants wore a red jacket for the duration of this study, and the robot detected red objects within a bounded rectangle in front of the robot's table using an overhead camera. The robot only checked for attention before starting each action, so it did not detect attention if the participant walked up to the robot in the middle of an action. This most closely follows the AMPS algorithm, as it assumes that the robot knows whether attention is present for an action before choosing it. The robot

also used this method to determine when to start actions when asking for attention; after requesting attention, the robot waits until it detects a red object to begin the action. When attention was detected, the robot said "I see you are here to give feedback."

After each algorithm ran, participants answered the following survey questions. Each question could be assigned a number from 1 (low) to 4 (high).

1. How **well** did the robot learn the task?
2. How **quickly** did the robot learn the task?
3. How **annoying** was the robot during the task?

These questions compared the algorithms' performance from the participants' perspectives. As AAMPS and AMPS Interval rely on interruptions, I measure how annoying each algorithm is perceived, hypothesizing that a robot that asks for less attention is perceived as less annoying. After all three algorithm rounds were completed, the participants gave their age, gender, robotics experience (1 low, 3 high), and answered two free-answer questions:

1. How would you improve the interface to make the interaction with the robot more effective?
2. How would you change the training process to make it easier to use or more effective?

The question regarding the interface was collected for future research on this topic, to determine whether the "good" and "bad" feedback buttons and verbal calls to the participant could be improved. This question is not meant for comparison between algorithms, rather to inform the methods that I used to collect feedback and alert people that the robot wants feedback.

### 4.2.5 AAMPS Robot Results: People have more break time with AAMPS

Participants self-reported a robotics background average of 1.5 out of 3, with two participants reporting at 3, so most of the users were inexperienced with robotics.

I measured the number of actions that received feedback from participants over each algorithm. Participants gave feedback on fewer actions during AAMPS ($M = 3.0, SD = 0.58$) than AMPS ($M = 5.83, SD = 2.11$) and AMPS Interval ($M = 4.0, SD = 0.00$). During AAMPS, participants gave 48.5% less feedback than AMPS, and 25% less feedback than AMPS Interval. The differences between these algorithms were statistically significant ($F(2, 31) = 3.38, p < 0.05$) using a repeated measures ANOVA. Post-hoc tests using a dependent t-test show that the difference between AAMPS and AMPS was statistically significant ($t(11) = -4.44, p < 0.001$), as were the differences between AAMPS and AMPS Interval ($t(11) = -5.74, p < 0.001$) and AMPS and AMPS Interval ($t(11) = 2.88, p < 0.05$). The amount of feedback per algorithm is shown in Figure 4.12.

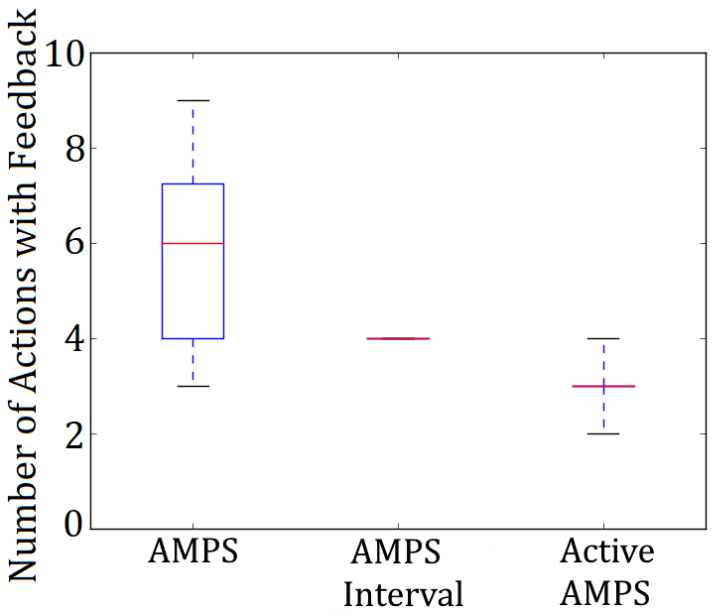



Figure 4.12: Amount of feedback given during each algorithm.

I measured the number of words written by participants over each algorithm. Words that were crossed out and rewritten were only counted once. Participants were able to copy more words during AAMPS ($M = 136.25, SD = 35.4$) than AMPS ($M = 76.75, SD = 36.5$) and AMPS Interval ($M = 120.17, SD = 29.0$). During AAMPS, participants completed 77.5% more work than AMPS, and 13.4% more work than AMPS Interval. The differences between these algorithms were statistically significant ($F(2, 31) = 3.55, p < 0.05$) using a repeated measures ANOVA. Post-hoc tests using a dependent t-test show that the difference between AAMPS and AMPS was statistically significant ($t(11) = 4.57, p < 0.001$), as were the differences between AAMPS and AMPS Interval ($t(11) = 2.65, p < 0.05$) and AMPS and

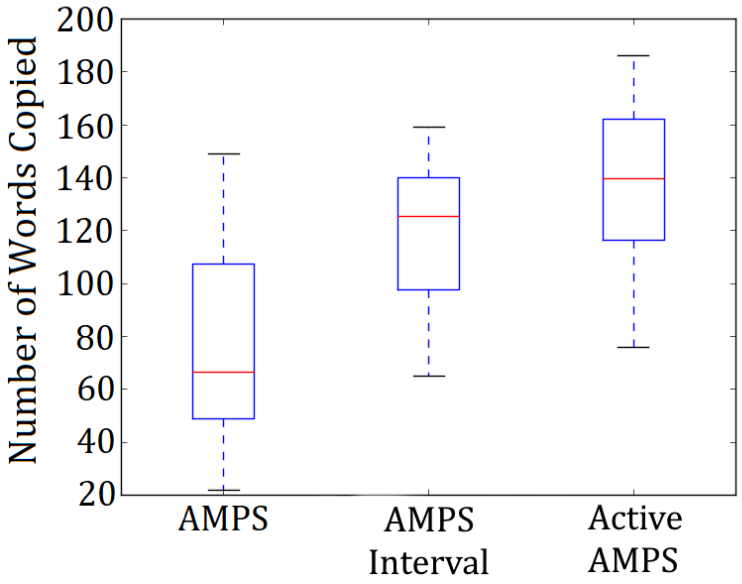



Figure 4.13: Amount of words written during each algorithm.

AMPS Interval ($t(11) = -4.26, p < 0.005$). The amount of words copied per algorithm is shown in Figure 4.13.

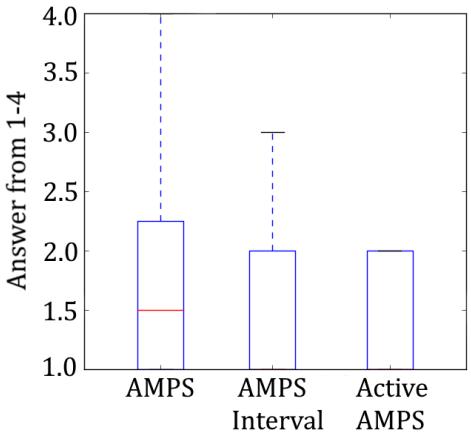
The participants were asked to report their annoyance with the robot, how well the robot learned, and how quickly the robot learned. All scores were on a scale from 1-4 (1 low, 4 high). Participants reported slightly less annoyance with AAMPS ($M = 1.33, SD = 0.47$) than AMPS ($M = 1.83, SD = 0.99$), but these differences were not statistically significant ($Z = 6.0, p = 0.16$) using the Wilcoxon Signed-Rank test. Similarly, they reported slightly less annoyance with AAMPS than with AMPS Interval ($M = 1.5, SD = 0.76$), but these differences were not statistically significant ($Z = 1.5, p = 0.41$). Ten out of twelve participants rated AAMPS as equally or less annoying than AMPS,

and eleven out of twelve participants rated AAMPS as equally or less annoying than AMPS Interval. The annoyance scores for each algorithm are shown in Figure 4.14a.

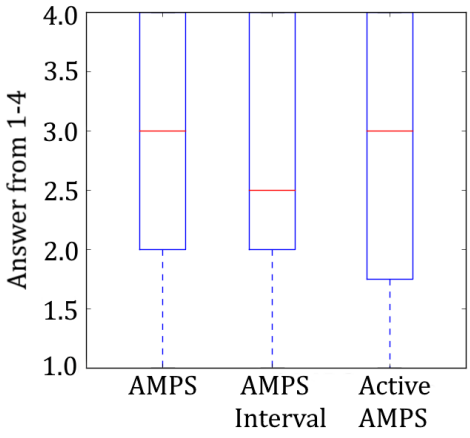
Participants reported that AAMPS ($M = 2.83, SD = 1.21$) learned better than AMPS Interval ($M = 2.67, SD = 1.11$), but these differences were not statistically significant using the Wilcoxon Signed-Rank test ($Z = 16.0, p = 0.78$). AAMPS was scored as not learning as well as AMPS ($M = 2.92, SD = 0.95$), but these differences were also not statistically significant ($Z = 12.0, p = 0.73$). The reported scores for each algorithm are shown in Figure 4.14b.

Participants reported that alg ($M = 2.0, SD = 1.0$) did not learn as quickly as AMPS ($M = 2.42, SD = 1.11$), but was not statistically significant using the Wilcoxon Signed-Rank test ($Z = 5.0, p = 0.24$). AAMPS was also reported to not learn as quickly as AMPS Interval ($M = 2.25, SD = 0.83$), but these differences were also not statistically significant ($Z = 13.5, p = 0.52$). The reported scores for each algorithm are shown in Figure 4.14c.

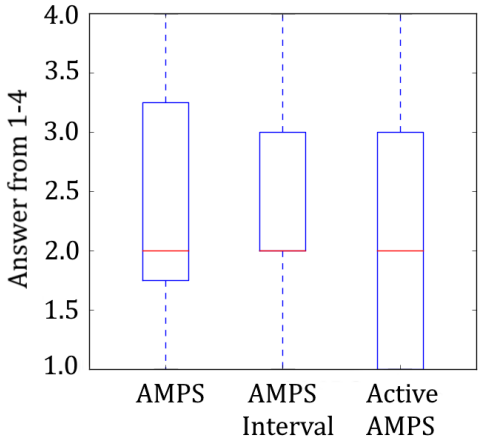
The answers to "How would you improve the interface to make the interaction with the robot more effective?" and "How would you change the training process to make it easier to use or more effective?" had several common themes. Participants suggested some technical improvements such as speeding up the robot or allowing feedback on partially observed actions. Five participants suggested more nuanced feedback techniques. For example, the ability to "rate...from 1 to 3 instead of good and bad" or to "help the robot




(a) How annoying was the robot?




(b) How well did the robot learn?




(c) How quickly did the robot learn?

Figure 4.14: Participant perceptions of the robot

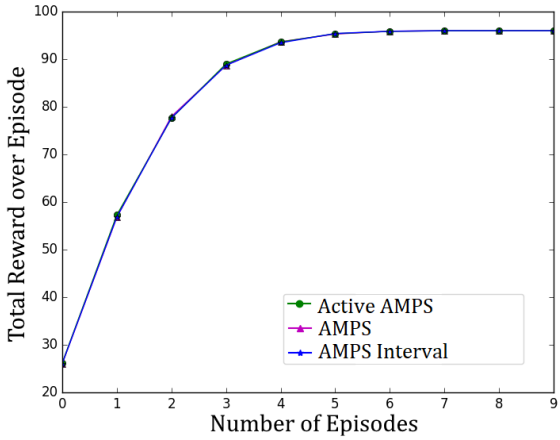



Figure 4.15: Simulated learning from human study data for each algorithm. The three algorithms learned at approximately the same rate.

know where to go, or prevent it from making an incorrect action." Three participants suggested ways to make it easier to multitask from the word copying station, including the ability to give feedback remotely so that they did not have to switch task stations. One participant suggested that the robot "make a sound when the action is completed" to help keep tabs on the robot's learning.

After the study, the final Q-values learned from each participant and algorithm were passed to the simulation, continuing learning in each algorithm without any feedback. The simulation averaged over 1000 trials of learning for 10 episodes from each participant's final Q-values for each algorithm. I found no statistically significant difference in the areas under these learning curves using Welch's t-test, between AAMPS ($M = 761.8, SD = 118.8$), AMPS Interval ($M = 761.0, SD = 119.5$), and AMPS ($M = 761.4, SD = 119.1$). The learning curves after the human study are shown in Figure 4.15.

## 4.3 Summary: AMPS and AAMPS improve the performance of HRL with inattentive teachers

My AMPS results suggest that the average area under the AMPS learning curve is consistently higher than the average area under the PS learning curve. Therefore, after the person stops paying attention to the robot and leaves the room, the robot can be expected to perform better on average using AMPS over PS. The lower variance in the average area under the AMPS curve may allow more trust in the learning algorithm overall, as it provides more consistent performance.

In Figure 4.6b, AMPS and PS perform similarly during both attention and inattention. One would only expect AMPS to outperform PS on average during inattention during early rounds, as shown in simulation, and the first period of inattention is short. The difference between the two algorithms can be seen in Figure 4.6a, which shows the second longer period of inattention. Figure 4.7 suggests we would see better performance from both AMPS and PS given more feedback from users. AMPS may also be sensitive to the amount of feedback given, as the more positive feedback it has received, the longer the robot will be able to act without trying new and unseen actions.

AMPS may more closely match people's expectations of how the learning process should proceed. The robot prioritizes actions that add and confirm task knowledge while the human teacher is present, and prioritizes listening to prior positive feedback while no teacher is present. This behavior is similar to social referencing, which serves a role in human development by allowing

infants to explore new actions while looking to a trusted authority for feedback [25].

In the AAMPS study, although I did not find significant results for how annoying participants found the robot during the different algorithms, the participants wrote fewer words and gave more feedback to the robot during AMPS and AMPS Interval. This shows that people spent more time with the robot than necessary when they chose how to divide their time or when the robot asked for attention at regular intervals. Thus even if people do not find the other algorithms annoying, AAMPS is able to better manage the participants' time. Although no algorithm learned significantly faster than the others in the human study, I attribute this to the short amount of time people interacted with the robot (twelve actions per algorithm).

Furthermore, note that the reported annoyance scores skewed towards low numbers. There are a variety of possible reasons for this result, including the short time period of the study, low experience with robots, and that none of the algorithms were extremely annoying. Thus differences in annoyance levels may be difficult to detect in a short-term study, but might be detected when people teach robots over long periods of time. I did not find significant results for how well and quickly the robot learned the task, suggesting that asking for feedback more or less often does not affect how intelligent the robot seems. How well or quickly the robot seems to learn could hinge on the random choice between taking a "good" or an "unseen" action while a teacher is watching.

The open-ended requests to make multi-tasking easier add to the evi-

dence supporting the benefits of the robot choosing when the teacher should pay attention. In addition to the increase in words copied and decrease in attention to the robot shown in AAMPS, the open-ended feedback suggests that people find multi-tasking while teaching a robot difficult.

The results for AMPS and AAMPS suggest that using these algorithms will allow robots to learn effectively from human teachers who need to take breaks. They may more closely match people's expectations of how the learning process should proceed, by adding and confirming task knowledge while the human teacher is present, and listening to prior positive feedback while no teacher is present, much like social referencing behavior in infants [25]. AMPS learns more quickly than Policy Shaping by effectively directing human attention to useful states. AAMPS allows robots to learn more quickly than AMPS with less burden on human teachers, using significantly less feedback and faster learning. Applying AAMPS to long-term learning will enable human teachers to be more productive, and allow robots to learn tasks more quickly by receiving feedback on important states, with less burden on teachers to determine which states are important.

# Chapter 5

# Learning from Incorrect Teachers

> *"I am at a rough estimate thirty billion times more intelligent than you. Let me give you an example. Think of a number, any number." ... "Wrong. You see?"*
>
> —Marvin the Paranoid Android (Douglas Adams, The Hitchhiker's Guide to the Galaxy)

As introduced previously, input from external sources is not always correct. Sensors can intermittently fail, and teachers can be wrong. It is difficult for robots to predict ahead of time exactly how often, and in what ways, advice or feedback will be incorrect. Often, the human teacher or other source of incoming feedback is unmodeled. If robots can use task information to model when and how feedback is incorrect, they can ignore incorrect feedback and learn from correct feedback.

Human-in-the-loop Reinforcement Learning (HRL) enables agents to learn from two sources: rewards taken from observations of the environment, and feedback or advice from a secondary critic source, such as human teachers or sensor feedback. The research in this chapter is based on the insight that the quality of the feedback can depend on the state-action pair. For example, consider a vision system that loses sight of objects outside a certain range, or a human teacher who gets confused about what the goal of the task is. There

are many situations in which incorrect feedback is structured, not random, and these algorithms exploit this structure. The result is algorithms that are robust to a wide variety of feedback correctness, and do not require prior knowledge of the teacher.

[1]

## 5.1 Revision Estimation from Partially Incorrect Resources

In this work, I present an algorithm, Revision Estimation from Partially Incorrect Resources (REPaIR), which can learn from incorrect teachers as defined in Section 3. REPaIR estimates corrections to imperfect feedback over time, acting as a feedback filter for RL algorithms with a reward function and additional environmental feedback, so that the quality of feedback does not need to be known ahead of time. REPaIR takes advantage of problem formulations in which the robot has access to a correct reward function, which is important since it has been shown that without some kind of additional information there is no way to recover from an incorrect reward signal [23].

REPaIR uses the cumulative reward at the end of each learning episode to determine whether the feedback received was correct or incorrect. If feedback received reflects the cumulative reward received (positive feedback leads to higher total reward, negative feedback leads to lower total reward), the

[1] Parts of this chapter have been submitted for publication at the time of writing.

robot has a higher trust in the feedback. Otherwise, the robot has lower trust in the feedback. REPaIR then either keeps, discards, or changes the feedback based on the trust. REPaIR can be used to estimate feedback correctness for feedback-utilizing HRL algorithms.

I test REPaIR with three baseline algorithms: Policy Shaping (PS) [30], and two versions of TAMER+RL (TAMER-P, TAMER-W) [38], which all have trust parameters to handle varying feedback quality. I show that adding REPaIR to these baselines matched or exceeded performance for 83.33% (TAMER-P), 83.33% (TAMER-W), and 100% (PS) of tested feedback quality settings in simulation. The average performance with REPaIR in these settings exceeded or matched more than half of mismatched trust parameters, which implies that REPaIR matches or improves expected performance on these baselines when they do not have prior information on feedback quality. I also demonstrate REPaIR on a manipulation task with a physical robot in which the robot must grasp a cup, with feedback provided by a noisy object detector. I found the differences in means consistent with the results in simulations. [2]

### 5.1.1 REPaIR Methodology: Incorporating feedback filtering into HRL

In this work, the algorithm takes advantage of a correct reward function $R$. Otherwise, with an imperfect teacher and without additional information,

[2]Parts of this section have been previously published under *Interactive Reinforcement Learning with Inaccurate Feedback [35] ©2020 IEEE.*

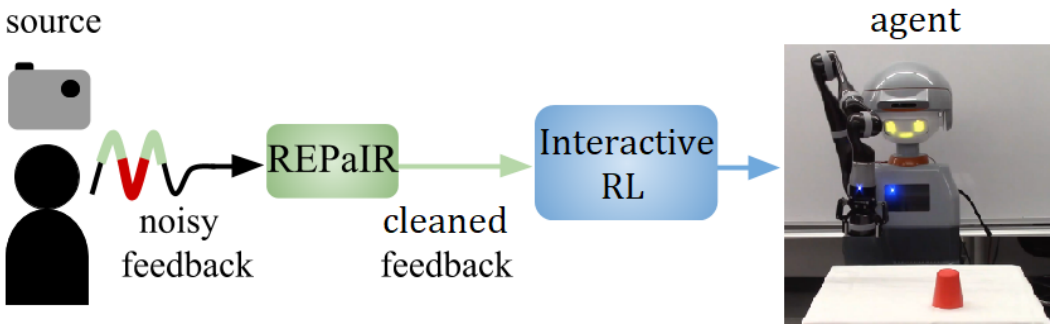



Figure 5.1: REPaIR Framework

it is impossible for the agent to tell what behavior is actually desired, as shown in [23]. I use sparse reward functions, as consistent feedback is most helpful when the reward function is not meticulously shaped to guide the agent to the goal. $R$ has positive rewards only on goal states and low negative rewards only on states to be avoided, potentially with small negative rewards on all other state-action pairs if fast travel to goal states should be encouraged. Such reward functions are easy for people to define, compared to dense reward functions. However, they are also more difficult for agents to learn from than well-populated rewards, as high-magnitude rewards will take longer to propagate through the large areas of small negative reward. Thus $F^*$ acts as a dense representation of $R$, as the agent can receive meaningful ranked feedback on each state-action pair, even in early learning when information has not yet propagated through the value function.

REPaIR leverages a correct reward function $R$ to compensate for incorrect feedback. To do so, I observe that the current state and action has some impact on the correctness of the feedback. Some prior work has assumed

that feedback either improves or worsens over time, or that feedback is randomly incorrect some percentage of the time [30, 38]. In practice, the current state-action pair often has an effect on the correctness of feedback. For example, consider a human teacher that does not understand the joint limits of a robotic arm. In this case, states and actions near the limits can receive incorrect feedback, as the teacher may give feedback that suggests an efficient path that takes the robotic arm into unsafe or impossible joint positions. However, other states and actions receive correct feedback. Another example is a vision system that can reliably detect distinct objects, but not if two such objects are too close to each other. In this case, states in which two or more objects are close may receive incorrect feedback on actions intended to grasp one object.

I present Revision Estimation from Partially Incorrect Resources (REPaIR) in Algorithm 3. REPaIR gives an estimation of the inverse of the corruption function, $\Gamma$ (as defined in Section 3), to an HRL algorithm to compensate for bad feedback. I will refer to this estimation as $\Gamma'$, where the input to $\Gamma'$ is $F(s, a)$, a state-action pair $(s, a)$ with feedback $f$. The output of $\Gamma'$ is an estimated revision $f'$, an attempt to recreate $F^*$. $\Gamma$ cannot be directly measured, as there is no ground truth for correct feedback until the value function is learned, at which point revisions to feedback are unnecessary. $\Gamma'$ gives corrected feedback to a learning algorithm.

To update $\Gamma'$ (and thus its estimate of the inverse of $\Gamma$), the agent uses the only ground-truth feedback, the reward function $R$. Since immediate high rewards do not always indicate the highest cumulative reward when the goal

is reached, especially in sparse reward functions, cumulative rewards collected at the end of each episode are used as ground-truth information.

As the agent learns, it saves the state-action trajectory $\xi$ that it takes in each learning episode, with feedback/advice $f_i$ for each $(s_i, a_i)$. At the end of each episode, it saves its final total reward, $R_\xi$. For $(s_i, a_i, f_i) \in \xi$, if a higher $R_\xi$ has not been seen for $(s_i, a_i, f_i)$, it is saved in the highest rewards: $R_{max}[(s_i, a_i, f_i)] = R_\xi$. The feedback on $(s_i, a_i, f_i)$ is assigned a trust $t_i$ in the range $[0, 1]$:

$$t_{(s_i,a_i,f_i)} = \begin{cases} \frac{R_{max}[(s_i,a_i)] - min(R_{max})}{\max(R_{max}) - \min(R_{max})} & f_i \text{is positive} \\ 1 - \frac{R_{max}[(s_i,a_i)] - min(R_{max})}{\max(R_{max}) - \min(R_{max})} & f_i \text{is negative} \end{cases} \tag{5.1}$$

The intuition behind this trust assignment is that an action should not be catastrophic if it results in one of the higher seen cumulative rewards. REPaIR determines whether to invert, keep, or discard feedback as follows, where $t_{min}$ and $t_{max}$ are threshold parameters. If $t_{(s_i,a_i,f_i)} \geq t_{max}$ , REPaIR keeps the feedback: $f_i = f_i$. If $t_{(s_i,a_i,f_i)} \leq t_{min}$, REPaIR inverts the feedback: $f_i = -f_i$. Otherwise, REPaIR discards the feedback: $f_i = 0$.

### 5.1.2 REPaIR Simulation Experiment: Comparing average reward gathered against baselines

In these experiments, I compare against three baselines and Q-Learning [84] (RL without feedback): two versions of TAMER+RL [38], and Policy Shaping (PS) [30]. REPaIR is used to supplement the two TAMER+RL

**Algorithm 3:** REPaIR

$\Gamma'$ = feedback revision estimator;
$R_{max} = -\infty$ = maximum total rewards seen; $t_{min}, t_{max}$ = thresholds for inverting and keeping feedback;
**while** *learning* **do**
  $R_\xi = 0$;
  $\xi = []$;
  **while** *episode not over* **do**
    $s_i, a_i$ = current state, action;
    $R_\xi = R_\xi$ + reward;
    $f_i$ = feedback;
    $\xi$.append($[s_i, a_i, f_i]$)
  **end**
  **for** $(s_i, a_i, f_i) \in \xi$ **do**
    $R_{max}[(s_i, a_i, f_i)] = \max(R_{max}[(s_i, a_i, f_i)], R_\xi)$;
    $t_{(s_i,a_i,f_i)} = \begin{cases} \frac{R_{max}[(s_i,a_i)] - min(R_{max})}{\max(R_{max}) - \min(R_{max})} & f_i \text{is positive} \\ 1 - \frac{R_{max}[(s_i,a_i)] - min(R_{max})}{\max(R_{max}) - \min(R_{max})} & f_i \text{is negative} \end{cases}$;
    **if** $t_{(s_i,a_i,f_i)} \leq t_{min}$ **then**
      $\Gamma'(f_i) = -f_i$;
    **else if** $t_{(s_i,a_i,f_i)} \geq t_{max}$ **then**
      $\Gamma'(f_i) = f_i$;
    **else**
      $\Gamma'(f_i) = 0$;
    **end**
  **end**
**end**

methods and PS. I chose these baseline algorithms because they take additional environmental feedback but do not use it to modify their reward or value functions, which can cause changes to the final optimal policy [55]. Thus the reward function will remain correct, which is a requirement for REPaIR. I expect REPaIR to improve the performance of these baselines because REPaIR estimates feedback quality based on the current state and action, rather than assuming a static trust [30] or simply decreasing trust over time [38]. Thus, the trust parameters for these baselines do not need to be varied to match the feedback quality, which may not be known in advance in a real-world scenario. I implemented all algorithms in Python 2.7.

For an RL baseline, I use Q-learning with Boltzmann exploration [83, 84]. More details can be found in Section 2.1.1. All of the following baselines are implemented with Q-Learning as the underlying RL algorithm. I compare REPaIR to TAMER+RL (detailed in Section 2.1.3) and Policy Shaping (detailed in Section 2.1.2). For TAMER+RL, $p$ and $w$ are annealed over time, as in [38]. They are decreased by 0.01% at the end of each learning episode.

I run experiments in simulation to compare the performance of HRL algorithms with and without REPaIR. I use a task in simulation for which it is straightforward to modify the feedback performance. The agent learns to place six objects into two bins, with four objects in bin one ($b_1$) and two objects in bin two ($b_2$). The agent has sixteen objects total, and can place or remove one object at a time from the bins. The agent can also choose to end the task at any time, and must do so to end the learning episode. The MDP

is as follows:

- $S$ : [$b_1$ contents, $b_2$ contents]
- $A$ : ["place one object into $b_1$", "place one object into $b_2$", "remove one object from $b_1$", "remove one object from $b_2$", "end task"]
- $T(s, a, s')$ : deterministic transition function
- $R(s, a)$ : +100 if $a$ = "end task" at goal state, -10 if $a$ = "end task" not at goal state, -1 otherwise.

I define perfect feedback ($F^*$) for this task as follows. The critic gives positive rewards for ending the task when there are four objects in $b_1$ and two objects in $b_2$, for adding objects to $b_1$ or $b_2$ when there are less than 4 and 2 respectively, and for removing objects from $b_1$ or $b_2$ when there are more than 4 and 2 respectively. The critic gives negative feedback otherwise.

For this task, I averaged over 50 trials of Q-Learning to optimize the parameters for area under the learning curve (the cumulative reward over all 100 episodes): $\tau = 0.5$, $\alpha = 0.8$, $\gamma = 0.8$. If the robot does not end the task before 100 actions, the episode ends with -10 reward.

I ran experiments adding REPaIR to three different algorithms in simulation: Policy Shaping (PS) [16, 30] and the top two performing algorithms from TAMER + RL [38] (TAMER-P, TAMER-W). The feedback quality is varied as follows. Some percent of states are chosen at random to receive correct feedback. The correct percentage is varied from 0 to 100, in increments

of 20. I compare the algorithm (PS, TAMER-P, TAMER-W) to Q-Learning and the algorithm with all feedback first given to REPaIR. All feedback is binary good/bad (1,-1) to maintain consistency across all algorithms. Thus in the TAMER feedback predictor (for state-action pairs where no feedback has been received), all positive feedback predictions are mapped to +1 and negative predictions to -1. All experiments are run over 100 learning episodes and averaged over 30 trials of each type of teacher. The results show the average area under the learning curve (the total reward over all 100 episodes) for each level of feedback correctness. This area is calculated using the composite trapezoidal rule.

One advantage of REPaIR is that it allows a single trust/weight parameter to be used across a wide variety of feedback reliability, rather than requiring tuning to a specific source of feedback. Therefore, all trust/weight parameters ($C$,$w$,$p$) are set to 0.8 for the experiments. I chose 0.8, as this value weights feedback positively but does not trust it fully. The trust parameters are varied for the baseline algorithms. This tests whether the addition of REPaIR outperforms the baseline algorithms if the quality of the teacher is not known ahead of time. Trust parameters are varied from 0.0 to 1.0 in increments of 0.1 for all baselines, with minor exceptions. For TAMER, $w = 0.0$ and $p = 0.0$ are equivalent to Q-Learning, so these are not tested. For PS, $C = 0.5$ is equivalent to Q-Learning, and $C$ cannot be exactly 0 or 1 as these lead to dividing by zero. Thus $C$ is set between 0.01 and 0.99, excluding 0.5. The results measure the percentage of trust settings (out of 10 total) that

perform differently than the baseline plus REPaIR.

I set $t_{min}$ and $t_{max}$ for each baseline algorithm. Recall that feedback is inverted when the trust is less than or equal to $t_{min}$, and kept when the trust is great than or equal to $t_{max}$. While these parameters are task- and algorithm-specific, they are not specific to the amount of incorrect feedback. For TAMER-P, TAMER-W, and PS, $t_{min}$ and $t_{max}$ are [0.05,0.85], [0.0,0.95], and [0.05,0.35] respectively.

### 5.1.3 REPaIR Simulation Results: REPaIR performs more dependably than baselines

In all results discussed, +REPaIR indicates that an algorithm was run with the feedback run through $\Gamma'$. All significance values are calculated using a one-way ANOVA and a Tukey post-hoc test, with $p < 0.05$ required for significance. In Figs. 5.2, 5.3, and 5.4, the two lines show performance for baseline+REPaIR and Q-Learning. The baseline algorithm is represented as a gradient of points, each of which represents one run (100 episodes long), where the color represents the trust parameter setting for that run (darker is higher).

Results for TAMER-P and TAMER-P+REPaIR are shown in Fig. 5.2. The percentage of $p$ settings for which TAMER-P+REPaIR significantly outperforms or underperforms the average TAMER-P is shown in Table 5.1. Across all feedback quality levels, adding REPaIR matched or exceeded baseline performance over the majority of $p$ settings in 83.33% of cases (starred in Table 5.1). TAMER-P+REPaIR also outperforms Q-Learning at 80% and

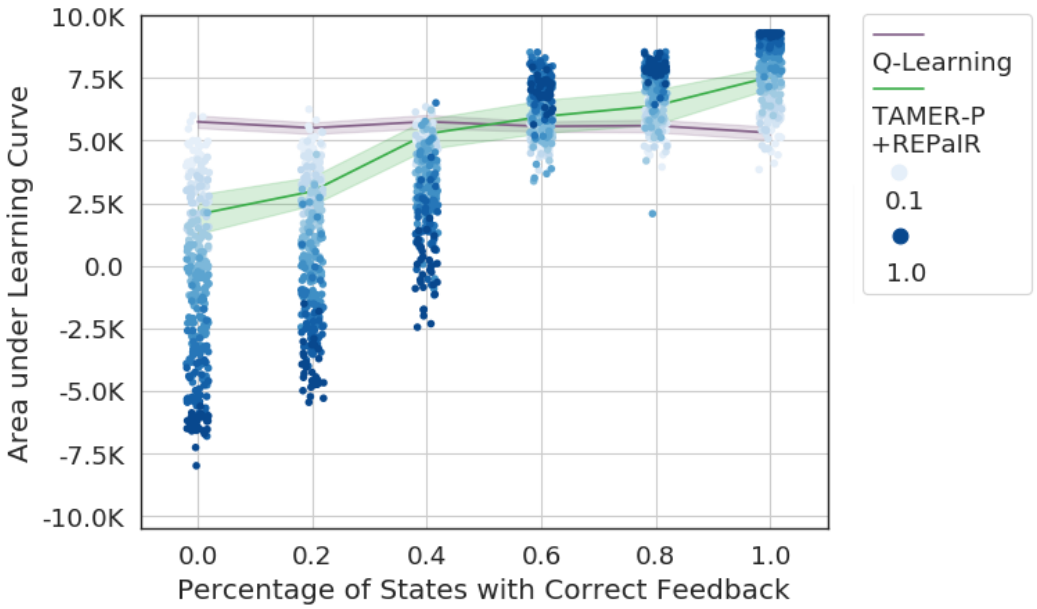



Figure 5.2: TAMER-P compared to TAMER-P+REPaIR and Q-Learning. The variable $p$ varies for TAMER-P as shown by the varied dots, and $p = 0.8$ for TAMER-P+REPaIR

| | $p$=0.0* | $p$=0.2* | $p$=0.4* | $p$=0.6* | $p$=0.8* | $p$=1.0 |
|---|---|---|---|---|---|---|
| Significantly Higher | **70%** | **70%** | **80%** | 0% | 0% | 30% |
| Similar | 10% | 10% | 20% | **60%** | **60%** | 20% |
| Significantly Lower | 20% | 20% | 0% | 40% | 40% | **50%** |

Table 5.1: Changes in performance from adding REPaIR to TAMER-P out of 10 different $p$ settings

100% correct state feedback.

Results for TAMER-W and TAMER-W+REPaIR are shown in Fig. 5.3. The percentage of $w$ settings for which TAMER-W+REPaIR significantly outperforms or under performs the average TAMER-W is shown in Table 5.2. Over all feedback quality levels, adding REPaIR matched or exceeded baseline

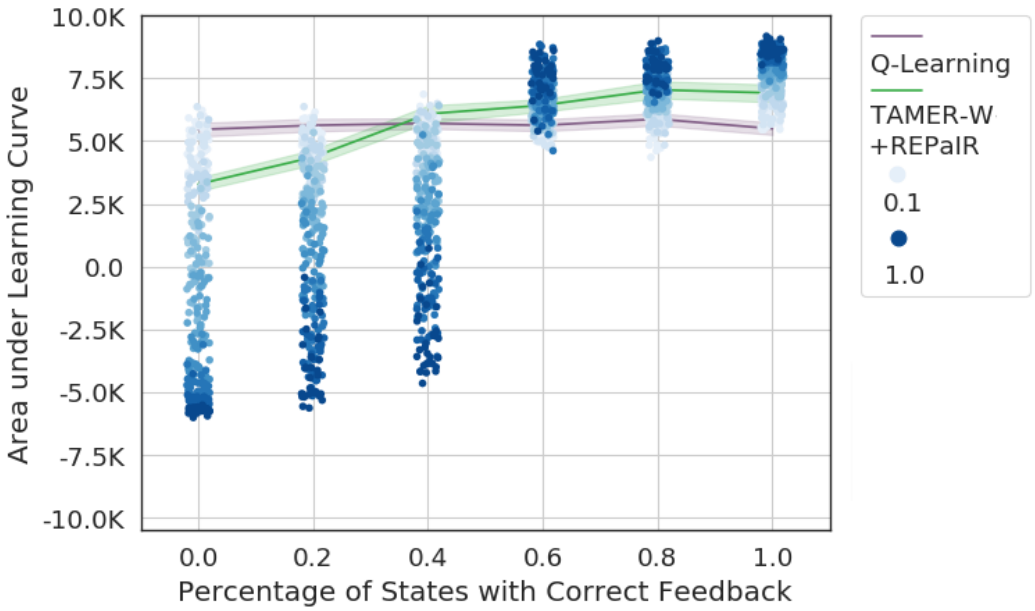



Figure 5.3: TAMER-W compared to TAMER-W+REPaIR and Q-Learning. The variable $w$ varies for TAMER-W as shown by the varied dots, and $w = 0.8$ for TAMER-W+REPaIR

| | $w$=0.0* | $w$=0.2* | $w$=0.4* | $w$=0.6* | $w$=0.8* | $w$=1.0 |
|---|---|---|---|---|---|---|
| Significantly Higher | **70%** | **70%** | **100%** | 30% | 30% | 10% |
| Similar | 10% | 20% | 0% | 30% | **40%** | 30% |
| Significantly Lower | 20% | 10% | 0% | **40%** | 30% | **60%** |

Table 5.2: Changes in performance from adding REPaIR to TAMER-W out of 10 different $w$ settings

performance over the majority of $w$ settings in 83.33% of cases (starred in Table 5.2). TAMER-W+REPaIR also outperforms Q-Learning at 60%, 80% and 100% correct state feedback.

Results for PS and PS+REPaIR are shown in Fig. 5.4. The percentage of $C$ settings for which PS+REPaIR significantly outperforms or under per-

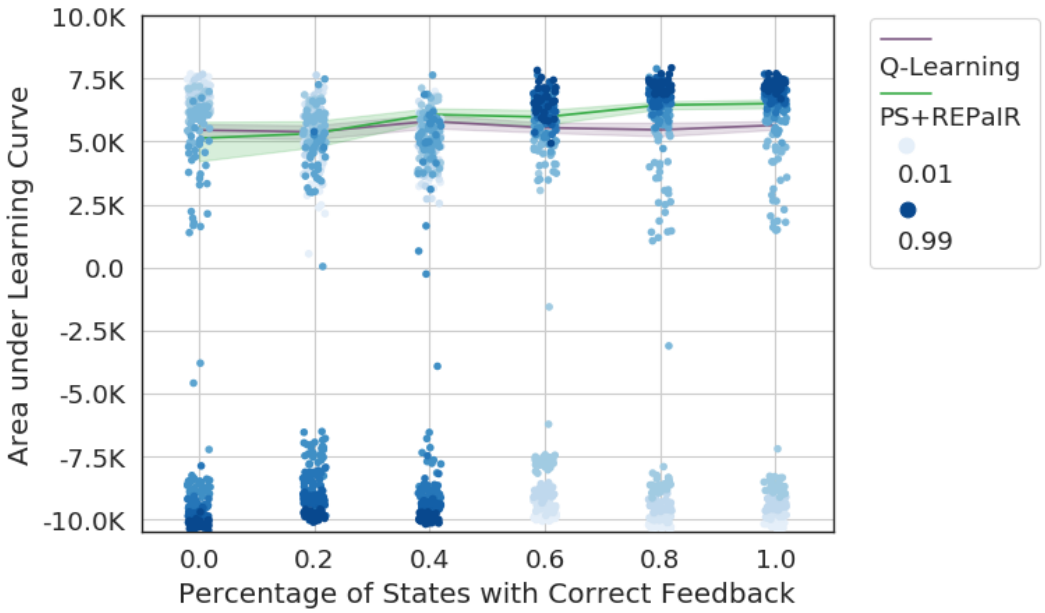



Figure 5.4: PS compared to PS+REPaIR and Q-Learning. The variable $C$ varies for PS as shown by the varied dots, and $C = 0.8$ for PS+REPaIR

| | $C$=0.0* | $C$=0.2* | $C$=0.4* | $C$=0.6* | $C$=0.8* | $C$=1.0* |
|---|---|---|---|---|---|---|
| Significantly Higher | **50%** | **60%** | **100%** | **50%** | **50%** | **60%** |
| Similar | 10% | 40% | 0% | 20% | 30% | 10% |
| Significantly Lower | 40% | 0% | 0% | 30% | 20% | 30% |

Table 5.3: Changes in performance from adding REPaIR to PS out of 10 different $C$ settings

forms the average PS is shown in Table 5.3. Over all feedback quality levels, adding REPaIR adding REPaIR matched or exceeded baseline performance over the majority of $C$ settings in 100.00% of cases (starred in Table 5.3). PS+REPaIR also outperforms Q-Learning at 60%, 80% and 100% correct state feedback.

### 5.1.4 REPaIR Robot Experiment

As a proof of concept, I also run experiments on a physical robot with a Kinova Jaco 7-dof arm and Robotiq gripper to compare the performance of Policy Shaping (PS) and PS+REPaIR. The agent learns to grasp a cup on a table by moving its gripper above the table in cardinal directions on a 4 by 4 grid, and reaching down to grasp when it is over the cup. The MDP is as follows:

- $S : x, y$ location of gripper in 4 by 4 grid
- $A$ : all four cardinal directions, and attempt a grasp
- $R(s, a)$ : +100 if robot successfully attempts grasp, -10 if robot unsuccessfully attempts grasp or after 16 actions, -1 otherwise.

The robot receives a reliable reward function from detecting whether its gripper is fully closed after attempting a grasp. If the gripper is not fully closed (blocked by the cup), the cup has been successfully grasped. The episode ends if the robot attempts a grasp or after a maximum of 16 actions. All experiments are 40 episodes long and averaged over 5 runs of each algorithm.

Feedback is given using the ORP object recognition and pose estimation system [5] to locate the position of the cup relative to the robot's gripper. If the gripper moves closer to the cup, the robot receives positive feedback, and receives negative feedback otherwise. When the arm gets in between the cup and the camera, the visual system may not perceive the cup and thus give

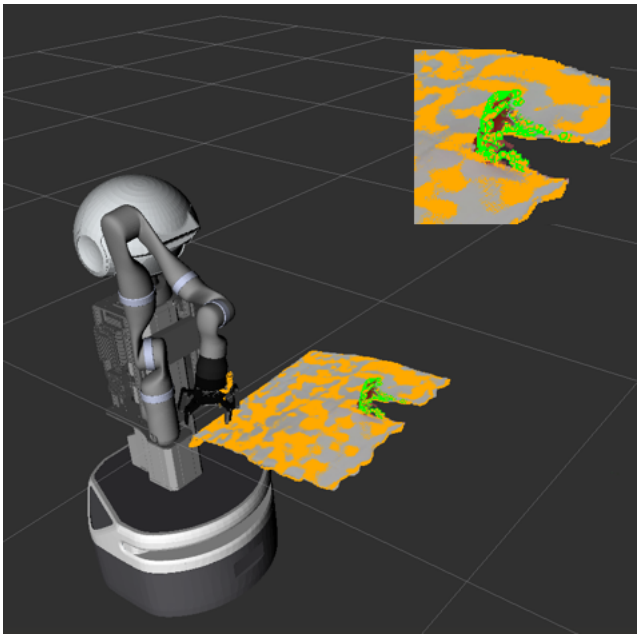


Figure 5.5: Robot vision using ORP [5] and Gazebo [39]

negative feedback, as the gripper is not getting closer to any perceived cup position. Other situations (such as the gripper intersecting the cup view) may give incorrect positions for the cup's location. When initially tested, the gripper fully blocked the cup in 25% of states, and partially blocked the cup in another 12.5%. Additional noise came from changes over time, such as changing indoor lights, movement, and arm positions. An image of the vision system is shown in Figure 5.5.

I optimized Q-Learning in simulation to optimize the parameters for area under the learning curve: $\tau = 0.1$, $\alpha = 1.0$, $\gamma = 0.9$. The experiment tests $C = 0.2$ $(PS - 0.2)$ and $C = 0.8$ $(PS - 0.8)$ as the midpoint performance of trusting or discounting all feedback. I set $t_{min}$ and $t_{max}$ to [0.0,0.95].

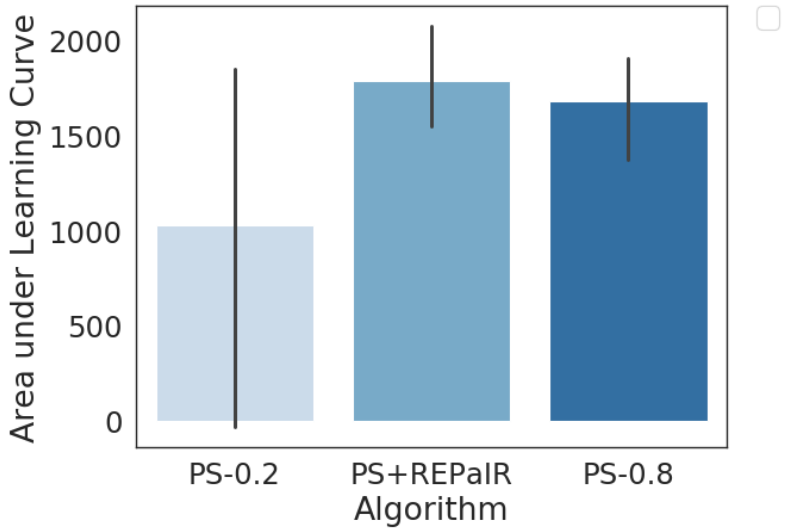



Figure 5.6: Performance of PS and PS+REPaIR on a robot

### 5.1.5 REPaIR Robot Results: Robot learns a task using REPaIR

Results are shown in Fig. 5.6. The addition of REPaIR, with an average of 1788.3 area under the learning curve, outperforms PS with $C = 0.8$, with an average of 1680.2 area under the learning curve, and PS with $C = 0.2$, with an average of 1026.7 area under the learning curve. These results are not significant ($p = 0.44$) using Welch's Anova, but show that a physical robot can learn a task using feedback filtered through REPaIR, and suggests that using REPaIR may improve performance.

## 5.2 Classification for Learning Erroneous Assessments using Rewards

This work focuses on extending the REPaIR algorithm to larger and potentially continuous state spaces. REPaIR was created for state spaces where every visited state-action pair can easily be stored in memory with the corresponding maximum cumulative reward. I present an algorithm, Classification for Learning Erroneous Assessments using Rewards (CLEAR), that, similar to the work by REPaIR, also uses achieved cumulative rewards to learn whether feedback is correct or incorrect over time. However, while REPaIR requires storing each observed state-action pair with an associated reward, which requires too much space and time for large state spaces, this work uses a classifier to store predictions of the slope of the learning curve based on observed state-action pairs, so each individual state-action pair does not need to be stored. CLEAR also adds in additional feedback as a supplement to the feedback from the human teacher.

Feedback is filtered through the CLEAR algorithm to a learning robot, and the performance of the feedback is explored in the beginning of the learning process. I test this algorithm against Policy Shaping [30] and Q-Learning with varying scenarios of human misunderstandings of a robot. The results suggest that using CLEAR as a feedback filter matches or exceeds the performance of Q-Learning over many levels of feedback quality, while the performance of Policy Shaping varies greatly based on feedback quality. This shows that CLEAR can help robots learn more dependably than baseline HRL methods.

### 5.2.1 CLEAR Methodology: Improving REPaIR with the use of machine learning

CLEAR uses an online learning classifier, $C_{CLEAR}$, to predict whether the RL learning curve is predicted to rise or fall based on state-action pairs. This information is combined with the environmental reward function $R$, which is assumed to be correct, to filter feedback. The predictions from $C_{CLEAR}$ are used to determine whether to keep, invert, or discard feedback, giving output similar to REPaIR. However, the method of choosing when to keep, invert, or discard feedback is different, and is achieved without thresholds that need to be set by an expert.

$C_{CLEAR}$ [3] takes in state features and actions, and outputs a prediction on whether the RL learning curve will rise, fall, or stay the same after the current trajectory. In general, a rising RL learning curve is a positive result, as the goal is to find the highest performing policy. Falling RL learning curves, in the absence of local minima, suggests a decrease in performance. CLEAR learns to predict the sign of the slope of the learning curve rather than learning the resulting scalar cumulative reward as is done in REPaIR. This problem would require regression and is a difficult problem to solve in larger and more complex state spaces.

CLEAR saves the state action pairs of the current episode's trajectory, $traj_e$, and the trajectory that has received the highest cumulative reward,

[3] In this specific work, $C_{CLEAR}$ is implemented as the multinomial Naive Bayes classifier from scikit-learn [58], with the training updates performed by the partial_fit() function.

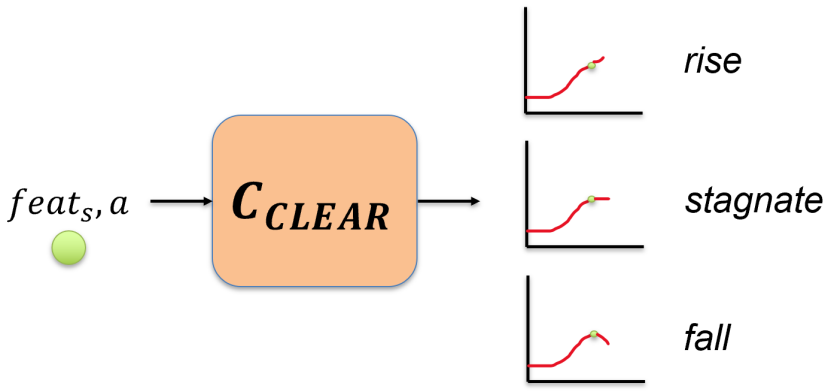



Figure 5.7: CLEAR algorithm: classifier for predicting learning slope

$traj_m$. The cumulative reward for episode $e$ is $R_e = \sum_{t=0}^{T} r_t$ where $T$ is the total number of time steps in an episode and $r_t$ is the reward received at each time step. The trajectories are composed of $(s_t, a_t)$ tuples, where $s_t$ is the state at time $t$, and $a$ is the action taken at time $t$. The current and maximum cumulative rewards are also saved: $R_e$, $R_{max}$. This information is used to preprocess data and determine whether the classifier should predict a rising, falling, or stagnating learning curve, as shown in Figure 5.7. $C_{CLEAR}$ takes as input the features of the state expected to influence the quality of human feedback (e.g. x-y gripper position, joint configuration, etc.) along with the action being taken. The output is one of three choices: *rise*, *stagnate*, or *fall*.

At the end of each trajectory, CLEAR trains $C_{CLEAR}$. Each training sample is given with a weight equal to the current episode count squared, weighting samples more heavily as learning continues and observed total rewards are more likely to be incorrect. $C_{CLEAR}$ as follows for each $(s, a) \in traj_e$,

where $feat_{s_t}$ is the features of state $s_t$:

- If $R_e > R_{max}$

$$C_{CLEAR}[(feat_{s_t}, a_t)] = \text{rise}$$

- Else if $R_e \leq R_{max} - (\text{episode\_count}/\text{max\_count}) * |(R_{max} - R_{min}))$

$$C_{CLEAR}[(feat_{s_t}, a_t)] = \text{fall}$$

- Else

$$C_{CLEAR}[(feat_{s_t}, a_t)] = \text{stagnate if } (s_t, a_t) \notin traj_m$$

A quick note that $C_{CLEAR}$ is trained with a "fall" label only if $R_e \leq R_{max} - (\text{episode\_count}/\text{max\_count}) * |(R_{max} - R_{min}))$, not simply $R_e < R_{max}$. I found this setting to work best in practice, to give a small but widening range to define *stagnation*, as the current episode count gets closer to the maximum number of episodes that the experimenter is running.

The feedback is then kept, discarded, inverted, or supplemented based on the predicted learning slope. That is, feedback is either directly passed through to the learning algorithm, not passed through at all, inverted by calculating $-\Delta_{s,a}$ as used in Policy Shaping (Section 2.1.2) [16,30], or added by CLEAR. When an HRL algorithm requests feedback, CLEAR is passed state-action pairs. CLEAR filters feedback by predicting the upcoming slope of the learning curve, $pred_s$. Using the predicted probabilities of $C$ [4], where $probs =$

[4]These probabilities were predicted using the predict_proba() function [58]

$C.predict\_proba([s, a])$, Pr[$pred_s$ = fall] = $probs$[fall], Pr[$pred_s$ = stagnate] = $probs$[stagnate], Pr[$pred_s$ = rise] = $probs$[rise]. The current feedback input is defined as $f$, and the total feedback received for a state is $\Delta_{s,a}$. Recall that Policy Shaping measures $\Delta_{s,a}$ as the difference in positive and negative feedback, so that a negative $\Delta_{s,a}$ means that the teacher disapproves of the state-action pair, and a positive $\Delta_{s,a}$ means that the teachers approves of the state-action pair. If there is no feedback ($\Delta_{s,a} == 0$), CLEAR supplements feedback $f$ proportional to the probability of the slope direction: $f = \frac{probs[\text{rise}]}{2}$ if $pred_s ==$ rise, and $f = -\frac{probs[\text{fall}]}{2}$ if $pred_s ==$ fall. Specifically, CLEAR returns feedback as follows (assuming feedback is input to a Policy Shaping baseline algorithm):

- If $\Delta_{s,a} < 0$ and $pred_s ==$ fall, or $\Delta_{s,a} > 0$ and $pred_s ==$ rise, KEEP: return $\Delta_{s,a}$
- Else if $pred_s =$ rise and $\Delta_{s,a} < 0$, or $pred_s =$ fall and $\Delta_{s,a} > 0$, INVERT: return $-\Delta_{s,a}$
- Else if $\Delta_{s,a} == 0$ and $pred_s ==$ rise, ADD: return $\frac{probs[\text{rise}]}{2}$
- Else if $\Delta_{s,a} == 0$ and $pred_s ==$ fall, ADD: return $-\frac{probs[\text{fall}]}{2}$
- Else, DISCARD: return 0.

If the baseline algorithm is not Policy Shaping, replace $\Delta_{s,a}$ with the feedback function for the baseline HRL algorithm.

### 5.2.2 CLEAR Simulation Experiment: Testing performance with simulated feedback

For these experiments, I compare CLEAR to baseline RL with no feedback, and to Policy Shaping with varying settings of the trust setting $C \in [0, 1]$. $C$ is set to 0.1, 0.25, 0.75, 0.9, and the true percentage of correct feedback $p*$. In the graphs, $C = p*$ is denoted by PS-1. For CLEAR, $C = 0.8$. For these experiments, the baseline algorithms use Q-Learning with Boltzmann exploration, with $\alpha, \gamma, \tau = 0.9$. The parameter $\tau$ is annealed by multiplying by 0.999 at the end of each episode. Each algorithm is run 100 times for a length of 750 episodes, with each episode ending after $2 * \sqrt{|S|}$, where $|S|$ gives the total number of states. The state features are the x and y coordinates of the robot gripper. A simulated teacher gives feedback to 80% of the robot's actions, chosen at random. The results are analyzed using a one-way ANOVA and Tukey post-hoc test.

The simulation task is a robot moving its gripper to touch a goal object on a flat surface. The discretized state space is 15x15 ($|S| = 225$, with the true goal at $(\sqrt{|S|} - 1, \sqrt{|S|} - 1)$. The robot arm can take cardinal and diagonal actions, one square at a time, or choose to not move. The robot arm starts at a random location at the beginning of each episode. There is also a distractor goal. In this task, the human confusion is pertaining to the goal; that is, out of two possible goals (the true and distractor), the teacher believes the distractor goal to be the true goal. As shown in 5.8, the distractor goal is placed at

1. (0,0)

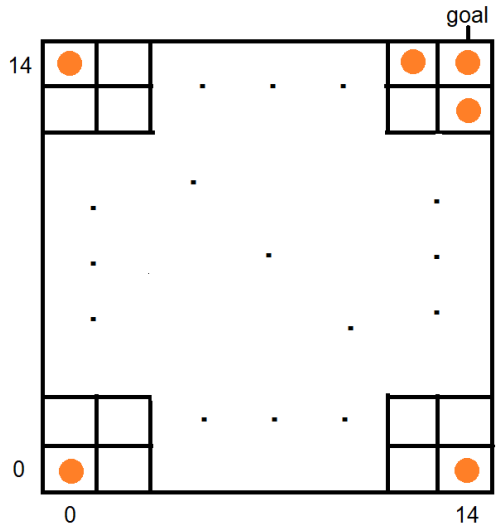



Figure 5.8: Distractor goal placements for $|S| = 225$

2. $(0, \sqrt{|S|} - 1)$
3. $(\sqrt{|S|} - 1, 0)$
4. $(\sqrt{|S|} - 2, \sqrt{|S|} - 1)$
5. $(\sqrt{|S|} - 1, \sqrt{|S|} - 2)$
6. $(\sqrt{|S|} - 1, \sqrt{|S|} - 1)$

The robot gripper starts at a random, non-goal and non-distractor-goal state at the beginning of each episode. The goal and distractor goal remain the same over each iteration. The simulated human feedback is modeled as an oracle that gives perfect feedback to the distractor goal. When the distractor goal is on the same space as the true goal, the simulated human gives perfect

feedback to the true goal. The simulated human gives feedback 80% of the time. The sparse reward function is:

- True goal: 10
- Distractor goal: 0.1
- All other states: -0.1

There is a reward placed on the distractor goal to show that even if there is a local maximum on the human's incorrect goal, CLEAR can recover.

### 5.2.3 CLEAR Simulation Results: CLEAR performs dependably over multiple levels of feedback correctness

The results of the simulation study are shown in Figure 5.9 and Table 5.4. I measured the $AUC$ using the composite trapezoidal rule for CLEAR, Q-Learning, the $C$ value that produces the minimum performing PS, and the $C$ value that produces the maximum performing PS, averaged over all 100 iterations. The average $AUC$ shows us the total rewards gather over time on average for each algorithm. Higher $AUC$s indicate that an algorithm achieved higher total rewards on average. Using a one-way ANOVA and a Tukey post-hoc test, CLEAR performs significantly better than Q-Learning ($p < 0.05$) for distractor placements ($|S| - 1$,0) and ($|S| - 1$,$|S| - 2$). However, the PS performance varies by large amounts based on the $C$ parameter. Since the simulated teacher gives feedback to an incorrect goal, without a model of how well the teacher performs, PS is subject to wide variances in performance.

This is likely due to the fact that the $C$ parameter, if matched properly to feedback quality, enables the robot to weight incoming feedback against state values appropriately. However, if the $C$ parameter and feedback quality are mismatched, PS will discount correct feedback too much, or put too much weight on incorrect feedback.

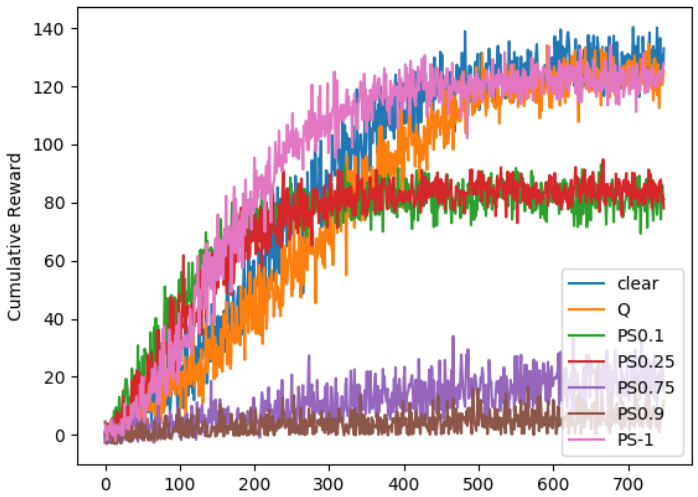
These results suggest that CLEAR performs more dependably than Policy Shaping with simulated human feedback. However, this does not test its performance with real people and messy feedback. I hypothesize that, even if trained on simulated human feedback, CLEAR could determine what feedback from real human teachers was correct or incorrect.

| DISTRACTOR | CLEAR | Q-LEARNING | PS-min | PS-max |
|---|---|---|---|---|
| (0,0) | 65290 | 59076 | 3015 | 70708 |
| ($\|S\|-1$,0) | 64528 | 53131 | 18426 | 42057 |
| ($\|S\|-1$,$\|S\|-2$) | 67682 | 56605 | 18970 | 44423 |
| (0,$\|S\|-1$) | 65626 | 5780 | 1376 | 88052 |
| ($\|S\|-2$,$\|S\|-1$) | 57267 | 53706 | 1415 | 481616 |
| ($\|S\|-1$,$\|S\|-1$) | 58851 | 52618 | 1211 | 124943 |

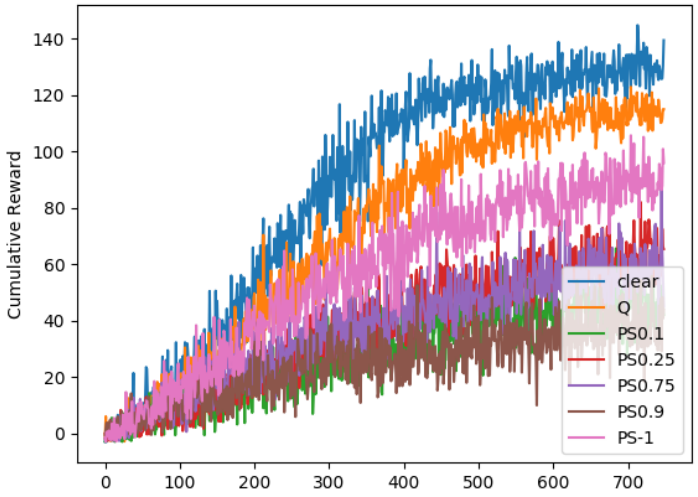
Table 5.4: The mean $AUC$ for each algorithm.

### 5.2.4 CLEAR Human Study: Testing how CLEAR responds to real human feedback

The human study for this work was run on Amazon Mechanical Turk with 10 participants. I used a simulated Fetch robot to run a modified version of the FetchReach-v1 task [12] that spans a 6x6 space over a table. This task is equivalent to the reach task used in the full simulation studies, except that the

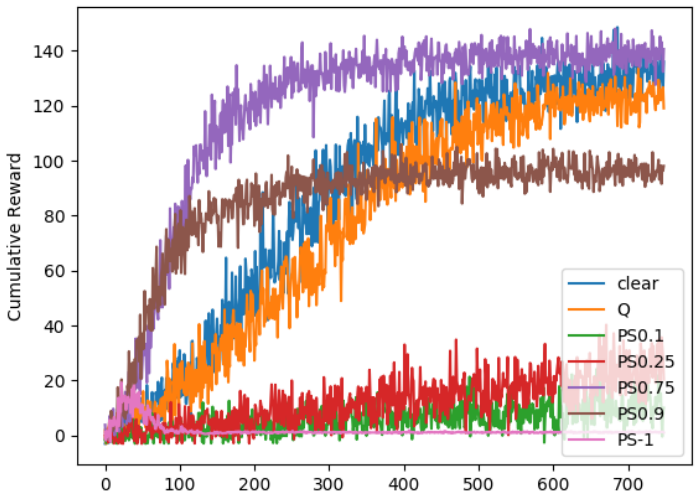



(a) Distractor goal (0,0)

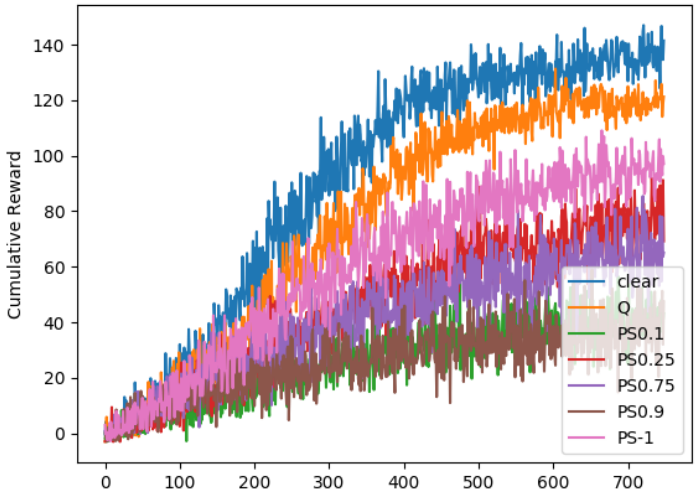



(b) Distractor goal ($|S| - 1$,0)

Figure 5.9: CLEAR simulation results with varied feedback correctness

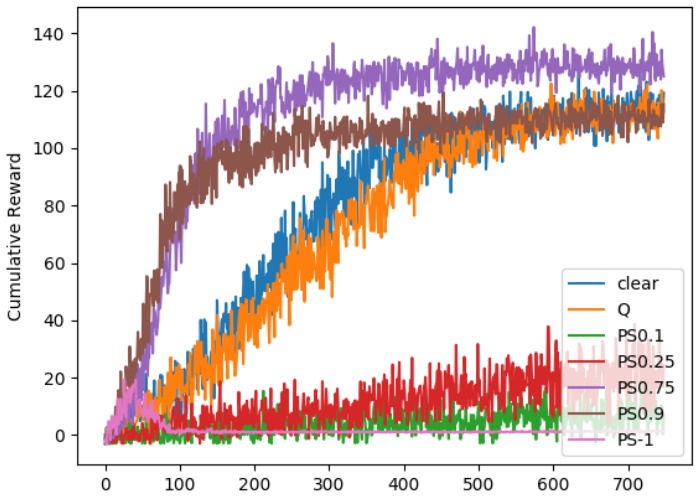



(c) Distractor goal ($|S|-1$,$|S|-2$)

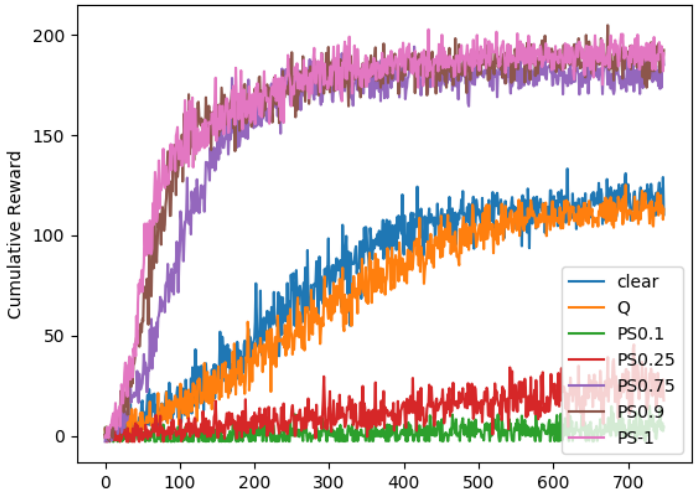



(d) Distractor goal (0,$|S|-1$)

Figure 5.9: Continued




(e) Distractor goal ($|S|-2$,$|S|-1$)




(f) Distractor goal ($|S|-1$,$|S|-1$)

Figure 5.9: Continued

robot begins at state (0,2) every time rather than a random starting location. The reward function is as follows:

- Goal state: +10
- Distractor state: +0.1
- Else: -0.1

The state features are the x and y coordinates of the robot gripper. The task still requires the robot to reach towards a goal object, with a distractor object present. This study began with a robot pretrained on simulated human feedback with a single true (2,1) and distractor goal object location (2,3). The Amazon Mechanical Turk participants viewed four videos of the robot reaching towards the different objects, taking the following paths (shown in Figure 5.10).

1. (0,2),(0,1),(1,1),(2,1)
2. (0,2),(0,3),(1,3),(2,3)
3. (0,2),(1,2),(2,2),(2,1)
4. (0.2),(1,2),(2,2),(2,3)

Each participant recorded their feedback for each action.

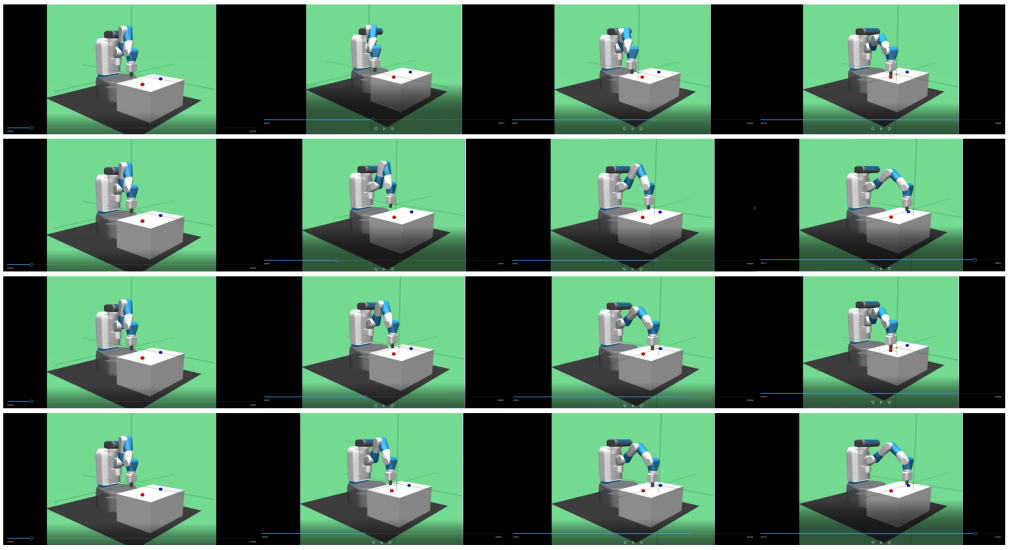


Figure 5.10: Videos in simulated robot environment, created using HIPPO Gym [77], MuJoCo [79], and OpenAI Gym [12]. Each row shows a video clip of the robot reaching to each goal (true and distractor) using two different trajectories.

### 5.2.5 CLEAR Human Study Results: CLEAR can filter messy human feedback

I obtained 100 classifier instances from running the CLEAR algorithm on the human study task with a simulated teacher, giving feedback to a blue distractor object. After obtaining the trained classifiers, I found the average performance of classifying the human data collected on Amazon Mechanical Turk. Each CLEAR episode ends after 7 actions. Although the participants were instructed to give correct feedback to the blue distractor object, the feedback was not clean, with some participants giving incorrect feedback to both the true and distractor object. Before filtering, the human data was 57.5% correct. After training, 65% of the kept feedback was classified correctly by CLEAR showing that, even trained on simulated incorrect data, CLEAR can improve human data through filtering.

The algorithm did discard some feedback, determining that it is of unknown quality. There were a total of 120 instances of feedback collected on Amazon Mechanical Turk. For each one of ten participants, we test over all 100 runs, and all 12 states visited in the human study. Thus we overall examine 1200 feedback instances. For these states over 100 runs, 3420 feedback instances were discarded, while 8580 were kept. CLEAR is able to learn more quickly than Policy Shaping and Q-Learning, as shown in Figure 5.11, which shows the pretraining in simulation to achieve fully trained classifiers before testing the human data. Furthermore, CLEAR keeps learning after PS and Q-Learning have settled on a sub-optimal policy, despite all the algorithms all

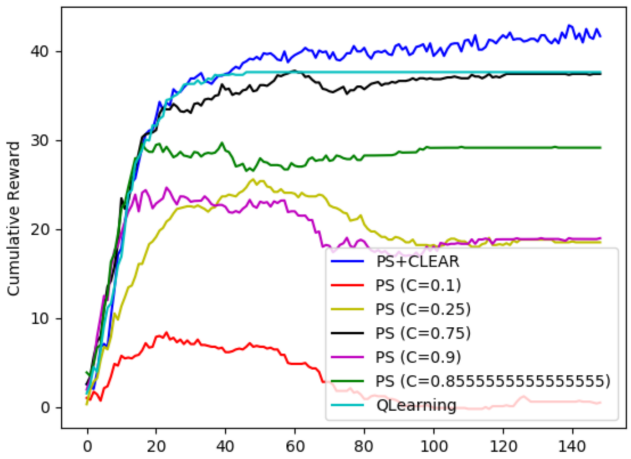



Figure 5.11: Simulated performance prior to Amazon Mechanical Turk data.

having the same learning rate parameter settings.

## 5.3 Summary: REPaIR and CLEAR perform more dependably than baselines

My results show that when HRL algorithms do not have prior knowledge on the correctness of a feedback source, using REPaIR to estimate better quality feedback improves performance. In practice, a robot will rarely know the quality of a feedback source in advance. A human teacher might have a wide range of understanding of a task, or they may act as an adversary. Sensor feedback might be highly useful for guiding learning, but might also be inconsistent in unpredictable ways. While REPaIR does have parameters that affect its performance, $t_{min}$ and $t_{max}$, these parameters can be set for a certain task and baseline RL algorithm, and do not rely on the correctness of the feedback. Furthermore, CLEAR does not require these parameter set-

tings, and can even add feedback to give to a learning robot. Thus REPaIR and CLEAR can be used to improve HRL performance when the quality of a feedback source is unknown.

The results for REPaIR and CLEAR suggest that using these algorithms will allow robots to learn effectively from human teachers who misunderstand a task and give incorrect feedback. These algorithms enable HRL to perform stably over varied human feedback. Both REPaIR and CLEAR perform similarly over different levels of feedback quality, while Policy Shaping and TAMER+RL are quite sensitive to the feedback quality. Applying these feedback filtering algorithms can improve the *dependability* of a learning robot when it does not know the quality of feedback ahead of time, which is likely to happen in the wild.

# Chapter 6

# Scalability of Presented Methods

*"Upgrading is compulsory"*
—Cyberman (Doctor Who: Series 2, Episode 5)

In this dissertation, I have presented four algorithms from learning from imperfect human teachers. All four algorithms were implemented and tested with Q-Learning as a baseline reinforcement learning algorithm. While Q-Learning works well on the kinds of tasks that I have studied throughout this thesis research, there are newer state-of-the-art reinforcement learning algorithms that can perform better on robots and with complex tasks, learning more quickly and robustly. In this chapter, I address the scalability of the methods I have proposed to more complex robotics tasks with state-of-the-art reinforcement learning methods. I first provide a brief background on some state-of-the-art reinforcement learning (RL) methods for robotics, covering Model-Based RL, Hierarchical RL, and Deep RL. These method are able to solve tasks to which basic Q-Learning does not scale well, such as continuous states and actions, large state spaces, and real-world robotics tasks.

Function approximation is one method of scaling RL to complex tasks [89], which enables agents to learn in continuous state and action spaces by

learning approximations of state and action spaces rather than storing all information directly. Neural Networks, modeled off of human brain function, are one such function approximation method that can be used with RL in continuous state and action spaces [1,47]. Deep Learning is the current state-of-the-art use of neural networks [60], which can also be applied to RL [3,8,81]. Given these developments in function approximation, there is a considerable body of HRL work that has expanded into deep learning. The baselines I work with in this dissertation have been expanded to deep learning: we created a Deep Policy Shaping (DPS) algorithm to learn in continuous spaces with discrete actions [85], and one of the methods of integrating TAMER signal with RL tested in [38] was extended to a Deep Q Network [6]. Several other HRL algorithms have also been implemented with deep reinforcement learning, such as Deep TAMER [82], Deep COACH [9], among others [45,72,87]. While these works extend HRL to Deep RL well, a potential downside of using Deep RL with human teachers is the amount of data needed for learning. Furthermore, when looking at continuous states and actions, human teachers may find it difficult to distinguish good actions from bad ones. For example, a teacher observing small changes in robot joint torques may find it difficult to provide feedback along the way.

Model-based RL is another approach to realistic RL on robotic tasks. Q-Learning, among other algorithms, falls under the umbrella of Model-free RL, which learns a policy directly from interacting with the environment. Model-based RL, on the other hand, uses a model of the world to predict what

might happen in future steps, and uses these predictions to explore and learn a policy [50, 90]. Model-based RL can lead to faster learning and less required robotics actions in the real world, as long as the model of the environment is accurate, making Model-based RL quite effective for robotics [59]. However, forming a model of a human teacher may be difficult. People can behave differently than expected and even change their behavior over time [78].

One other RL method that I will cover here is Hierarchical RL. Hierarchical RL enables scaling up to difficult tasks by dividing problems into subtasks, which also enables better task generalization [10, 57]. Hierarchical RL can divide large, complex tasks into manageable subtasks; for example, consider the task of setting a full table. While this may be a large task for a robot to learn with RL, Hierarchical RL can divide this task into placing plates, placing cups, placing silverware, and more. Those subtasks can be even further divided, into object grasping, placing, and the like. This subtask division can also improve transfer learning, or the ability to apply previously learned information to new tasks. If, after learning to set a table, you would like the robot to learn how to put away dishes, the robot can use previously learned subtasks, such as "pick up cup" and "place fork", and apply them to this new task.

My work on *inattentive* teachers – the AMPS and Active AMPS algorithms – can be extended to Deep RL, Model-Based RL, and Hierarchical RL in a fairly straightforward manner. While the baseline Policy Shaping method would need to change, the AMPS and Active AMPS algorithms focus

on changing the exploration of a robot using RL to learn. While there are many differences between state-of-the art RL algorithms, all of them use exploration in some way. Thus AMPS and Active AMPS can be used to guide the robot to explore more during attention and less during inattention. AMPS or Active AMPS may even lower the amount of feedback needed from users, potentially mitigating some of the concern over data requirements for deep learning. However, if using a non-Markovian RL method [28, 54, 69, 86], which could be designed as Deep, Model-Based, or Hierarchical RL, AMPS and Active AMPS may run into difficulties. In a Markovian task, teachers can become attentive, look at the current state of the robot, and be fully prepared to give feedback on future actions.

However, with non-Markovian tasks, previous actions may affect the future, no matter what the current state is. Thus, robots would have to bring teachers up-to-speed with the current state based on what they missed while being inattentive to the robot. There is work in swarm robotics on keeping human understanding current with the whole swarm at once, despite being distracted [66], how to make sure drivers and pilots are aware of the state when autopilot intervention is needed [29, 46], and how to ensure that people know enough about a robot's state to answer questions [64]. One of these methods could be integrated with AMPS and Active AMPS in order to apply them to a non-Markovian task.

My work in *inaccurate* teachers poses more difficulties when expanding to RL methods more complex than Q-Learning. REPaIR cannot extend di-

rectly to large, continuous state, or continuous action spaces, as the algorithm requires a memory of each observed state-action pair. Thus, extending directly to any learning method that performs well on complex tasks may be infeasible. REPaIR could potentially be applied to a Hierarchical RL method if only used for macro-actions (composed of multiple subtasks). However, CLEAR would likely still be a better choice than REPaIR.

CLEAR lends itself more easily to scalability. This algorithm could be applied to a Deep RL algorithm in a straightforward manner. To make CLEAR even more scalable, the $C_{CLEAR}$ classifier could be a neural net (or any other classifier). However, given the large amount of data required for deep learning, Deep RL may not be the best choice for HRL tasks with inexpert teachers. Model-based RL may be more feasible for extending CLEAR to complex tasks. Since the classifier can be used to get slope predictions even in previously unobserved states, $C_{CLEAR}$ could be used to provide a model of human feedback to supplement the environmental model. Observed feedback could still be used to update $C_{CLEAR}$. However, this extension would need to be tested, as CLEAR may receive less feedback given that Model-based RL enables robots to take fewer real-world actions. Finally, Hierarchical RL may also be a good state-of-the-art method for extending CLEAR. $C_{CLEAR}$ could be used to learn slope predictions for the entire task or for subtasks, and CLEAR would apply well to transfer learning as long as there are similar state features and action choices between each task. Since the state features and actions are the input to $C_{CLEAR}$, CLEAR could be used to predict feedback

on different tasks.

For all extensions, simply inverting the feedback as is done in CLEAR may not be the best option when feedback is distrusted. In more complex tasks, performance may be improved by instead simply decreasing the feedback applied to the algorithm; for example, $\Delta_{s,a}$ in Policy Shaping could be brought closer to 0, rather than starting by fully inverting to $-\Delta_{s,a}$ when the feedback is not trusted. Or, using learned information about the other actions, another action's $\Delta_{s,a_2}$ could be increased when feedback $\Delta_{s,a_1}$ cannot be trusted. What other forms inversions might take depend largely on the structure of the task and HRL algorithm as well as different kinds of inputs from teachers.

In this chapter, I have presented some ways in which my thesis could be expanded from using Q-Learning as a base algorithm to using newer, state-of-the-art RL methods. AMPS, Active AMPS, and CLEAR are the most scalable. Given that many HRL algorithms are Model-free or Deep RL algorithms, and my algorithms function by modulating exploration and filtering feedback to a baseline HRL algorithm, some modifications and future research may be needed to fully utilize my algorithms on complex robotics tasks. This chapter provides a road map for possible future extensions.

# Chapter 7

# Summary and Conclusions

*"Taking one last look, sir...at my friends."*
—C-3PO ( Star Wars: Episode IX - The Rise of Skywalker)

In this dissertation, I have proposed that robots using HRL should change the way they explore and learn based on human attention and errors when teaching. People can be effective robot teachers, but human behavior and performance is far from perfect; distractions, lack of time, and misunderstandings of the task or robot can impact human performance. Thus, the goal of this dissertation has been to show that:

**Actively modifying which states receive feedback from imperfect, unmodeled human teachers can improve the speed and dependability of Human-In-the-loop Reinforcement Learning (HRL).**

I presented my algorithms towards learning from ***inattentive*** teachers, Attention-Modified Policy Shaping (AMPS) and Active Attention-Modified Policy Shaping (AAMPS). While robots can still learn without attention, AMPS and AMPS allow robots to take advantage of human attention while

attempting to behave more optimally while unattended. The results suggest that modifying the state-action pairs observed by human teachers enables a robot to learn **faster** than when using a baseline HRL method. Furthermore, this change in learning based on human attention can allow for more breaks in teaching for human teachers, enabling a more natural and less tiring workflow than prior work.

Using AMPS, the average area under the learning curve is consistently higher than the average area under the PS learning curve. Therefore, after the person stops paying attention to the robot and leaves the room, the robot can be expected to perform better on average using AMPS over PS. The lower variance in the average area under the AMPS curve may allow more trust in the learning algorithm overall, as it provides more consistent performance.

I hypothesized that AAMPS would allow robots to learn as quickly as AMPS with less burden on human teachers, and that people would prefer being interrupted less often. The results suggest that AAMPS does learn more quickly than AMPS. Furthermore, AAMPS uses significantly less feedback than AMPS and Policy Shaping, showing that less feedback still results in fast learning when using AAMPS.

Secondly, I presented my work on learning from ***incorrect*** teachers, the algorithms Revision Estimation from Partially Incorrect Resources (REPaIR) and Classification for Learning about Erroneous Assessments using Rewards (CLEAR). My results suggest that when HRL algorithms do not have prior knowledge on the correctness of a feedback source, using REPaIR or CLEAR to

estimate better quality feedback improves performance. In practice, a robot will rarely know the quality of a feedback source in advance. REPaIR and CLEAR, by filtering the feedback that state-action pairs receive, enable robots to learn more **dependably** than baselines. Thus this work will allow for a wider range of skilled human teachers to successfully teach robots skills using HRL.

The average performance with REPaIR over different feedback qualities shows that REPaIR can improve performance when feedback quality is unknown. For TAMER-P and TAMER-W, REPaIR can decrease performance when feedback is perfect, but this is balanced by the substantial performance gains for imperfect feedback. This is because over-trusting feedback lets the robot take full advantage of correct feedback, but can lead to bad performance when feedback is bad. As shown by the color gradient of baseline points in Figs. 5.2, 5.3, and 5.4, the best performance comes from matching trust to actual feedback quality. In contrast, REPaIR can perform well with one trust setting, thus improving learning when the quality of feedback is not fully known in advance. I demonstrated the performance of REPaIR in a real-world robot experiment, which suggested that a robot can use REPaIR to filter feedback while learning.

The CLEAR results suggest that this algorithm performs more dependably than a baseline HRL algorithm in a medium size (225 state) space. While Policy Shaping can outperform CLEAR with the right parameters, there is often no way to model human teachers before the teaching process begins. Thus,

if you are picking a $C$ parameter before beginning, Policy Shaping may take quite some time to learn the task (much longer than RL with no feedback), whereas CLEAR does not perform significantly worse than RL in any of these results. This suggests that CLEAR is robust to a wide variety of inputs.

Together, this thesis lays the groundwork for an easier teaching experience for imperfect human teachers. The completed algorithms are steps towards enabling any person, unknown and unmodeled by the robot, to teach a new task in a comparable or shorter amount of time than prior algorithms, without needing to spend long consecutive hours watching the robot or be skilled at completing or teaching the desired task.

## 7.1 Contributions

To summarize, this thesis has provided the following contributions:

1. A HRL algorithm (AMPS) that changes RL exploration in order to learn significantly faster than a baseline with 44% higher area under the learning curve (**Speed**) [24] (Chapter 4)

2. A HRL algorithm (AAMPS) that enables robots to ask for attention from inattentive teachers when needed, performing significantly faster than baselines with $\geq 11\%$ higher area under the learning curve (**Speed**) [34] (Chapter 4)

3. A framework for Markov Decision Processes with incorrect feedback [35] (Chapter 3)

4. An algorithm (REPaIR) that filters imperfect feedback to various HRL algorithms, enabling the expected performance to match or exceed baseline when the robot has no prior model of expected human feedback correctness (**Dependability**) [35] (Chapter 5)

5. An algorithm (CLEAR) that filters imperfect feedback to a HRL algorithm in a large state space, as well as adding supplemental feedback, matching or exceeding the performance of an RL baseline, outperforming an HRL baseline when the robot has no prior model of expected human feedback correctness (**Dependability**)

# Bibliography

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad, *State-of-the-art in artificial neural network applications: A survey*, Heliyon **4** (2018), no. 11, e00938.

[2] Joshua Achiam and Shankar Sastry, *Surprise-based intrinsic motivation for deep reinforcement learning*, arXiv e-prints (2017), arXiv–1703.

[3] Forest Agostinelli, Guillaume Hocquet, Sameer Singh, and Pierre Baldi, *From reinforcement learning to deep reinforcement learning: An overview*, Braverman readings in machine learning. key ideas from inception to current state (2018), 298–328.

[4] Riad Akrour, Marc Schoenauer, and Michèle Sebag, *April: Active preference learning-based reinforcement learning*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2012, pp. 116–131.

[5] Adam David Allevato et al., *An object recognition and pose estimation library for intelligent industrial automation*, Master's thesis, University of Texas at Austin, 2016.

[6] Riku Arakawa, Sosuke Kobayashi, Yuya Unno, Yuta Tsuboi, and Shin-ichi Maeda, *Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback*, arXiv e-prints (2018), arXiv–1810.

[7] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning, *A survey of robot learning from demonstration*, Robotics and autonomous systems **57** (2009), no. 5, 469–483.

[8] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath, *Deep reinforcement learning: A brief survey*, IEEE Signal Processing Magazine **34** (2017), no. 6, 26–38.

[9] Dilip Arumugam, Jun Ki Lee, Sophie Saskin, and Michael L Littman, *Deep reinforcement learning from policy-dependent human feedback*, arXiv e-prints (2019), arXiv–1902.

[10] Andrew G Barto and Sridhar Mahadevan, *Recent advances in hierarchical reinforcement learning*, Discrete event dynamic systems **13** (2003), no. 1, 41–77.

[11] Steven Bird, Ewan Klein, and Edward Loper, *Natural language processing with python: analyzing text with the natural language toolkit*, " O'Reilly Media, Inc.", 2009.

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, *Openai gym*, 2016.

[13] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum, *Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations*, International conference on machine learning, PMLR, 2019, pp. 783–792.

[14] Allison Bruce, Illah Nourbakhsh, and Reid Simmons, *The role of expressiveness and attention in human-robot interaction*, Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, vol. 4, IEEE, 2002, pp. 4138–4142.

[15] Maya Cakmak, Crystal Chao, and Andrea L Thomaz, *Designing interactions for robot active learners*, IEEE Transactions on Autonomous Mental Development **2** (2010), no. 2, 108–118.

[16] Thomas Cederborg, Ishaan Grover, Charles L Isbell, and Andrea L Thomaz, *Policy shaping with human teachers*, Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[17] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh, *Intrinsically motivated reinforcement learning*, Advances in neural information processing systems, 2005, pp. 1281–1288.

[18] Sonia Chernova and Manuela Veloso, *Interactive policy learning through confidence-based autonomy*, Journal of Artificial Intelligence Research **34** (2009), 1–25.

[19] Jeffery Allen Clouse, *On integrating apprentice learning and reinforcement learning*, University of Massachusetts Amherst, 1996.

[20] Finale Doshi-Velez, Joelle Pineau, and Nicholas Roy, *Reinforcement learning with limited reinforcement: Using bayes risk for active learning in pomdps*, Artificial Intelligence **187** (2012), 115–132.

[21] Arkady Epshteyn, Adam Vogel, and Gerald DeJong, *Active reinforcement learning*, Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 296–303.

[22] Owain Evans, Andreas Stuhlmüller, and Noah Goodman, *Learning the preferences of ignorant, inconsistent agents*, Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[23] Tom Everitt, Victoria Krakovna, Laurent Orseau, and Shane Legg, *Reinforcement learning with a corrupted reward channel*, Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 4705–4713.

[24] Taylor Kessler Faulkner, Elaine Schaertl Short, and Andrea Lockerd Thomaz, *Policy shaping with supervisory attention driven exploration*, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 842–847.

[25] Saul Feinman, *Social referencing in infancy*, Merrill-Palmer Quarterly (1982-) (1982), 445–470.

[26] Mary Ellen Foster, Andre Gaschler, and Manuel Giuliani, *Automatically classifying user engagement for dynamic multi-party human–robot interaction*, International Journal of Social Robotics **9** (2017), no. 5, 659–674.

[27] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell, *Reinforcement learning from imperfect demonstrations*, arXiv e-prints (2018), arXiv–1802.

[28] Maor Gaon and Ronen Brafman, *Reinforcement learning with non-markovian rewards*, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 3980–3987.

[29] Jonas Gouraud, Arnaud Delorme, and Bruno Berberian, *Autopilot, mind wandering, and the out of the loop performance problem*, Frontiers in neuroscience **11** (2017), 541.

[30] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz, *Policy shaping: Integrating human feedback with reinforcement learning*, Advances in neural information processing systems, 2013, pp. 2625–2633.

[31] Daniel H Grollman and Aude G Billard, *Robot learning from failed demonstrations*, International Journal of Social Robotics **4** (2012), no. 4, 331–342.

[32] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband,

et al., *Deep q-learning from demonstrations*, Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[33] Mingxuan Jing, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Chao Yang, Bin Fang, and Huaping Liu, *Reinforcement learning from imperfect demonstrations under soft expert guidance*, AAAI, 2020.

[34] Taylor Kessler Faulkner, Reymundo A Gutierrez, Elaine Schaertl Short, Guy Hoffman, and Andrea L Thomaz, *Active attention-modified policy shaping: Socially interactive agents track*, Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 728–736.

[35] Taylor Kessler Faulkner, Elaine Schaertl Short, and Andrea L. Thomaz, *Interactive reinforcement learning with inaccurate feedback*, 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Submitted for review.

[36] W Bradley Knox and Peter Stone, *Tamer: Training an agent manually via evaluative reinforcement*, Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on, IEEE, 2008, pp. 292–297.

[37] W. Bradley Knox and Peter Stone, *Interactively shaping agents via human reinforcement*, Proceedings of the fifth international conference on Knowledge capture - K-CAP '09 (New York, New York, USA), ACM Press, 2009, p. 9.

[38] W Bradley Knox and Peter Stone, *Combining manual feedback with subsequent mdp reward signals for reinforcement learning*, Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 5–12.

[39] Nathan Koenig and Andrew Howard, *Design and use paradigms for gazebo, an open-source multi-robot simulator*, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3, IEEE, 2004, pp. 2149–2154.

[40] Samantha Krening and Karen M Feigh, *Interaction algorithm effect on human experience with reinforcement learning*, ACM Transactions on Human-Robot Interaction (THRI) **7** (2018), no. 2, 16.

[41] Samantha Krening and Karen M Feigh, *Newtonian action advice: Integrating human verbal instruction with reinforcement learning*, Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 720–727.

[42] Divesh Lala, Koji Inoue, Pierrick Milhorat, and Tatsuya Kawahara, *Detection of social signals for recognizing engagement in human-robot interaction*, arXiv e-prints (2017), arXiv–1709.

[43] Guangliang Li, Randy Gomez, Keisuke Nakamura, and Bo He, *Human-*

*centered reinforcement learning: A survey*, IEEE Transactions on Human-Machine Systems **49** (2019), no. 4, 337–349.

[44] Guangliang Li, Bo He, Randy Gomez, and Keisuke Nakamura, *Interactive reinforcement learning from demonstration and human evaluative feedback*, 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), IEEE, 2018, pp. 1156–1162.

[45] Zhiyu Lin, Brent Harrison, Aaron Keech, and Mark O Riedl, *Explore, exploit or listen: Combining human feedback and policy model to speed up deep reinforcement learning in 3d worlds*, arXiv e-prints (2017), arXiv–1709.

[46] Chen Lv, Dongpu Cao, Yifan Zhao, Daniel J Auger, Mark Sullman, Huaji Wang, Laura Millen Dutka, Lee Skrypchuk, and Alexandros Mouzakitis, *Analysis of autopilot disengagements occurring during autonomous vehicle testing*, IEEE/CAA Journal of Automatica Sinica **5** (2017), no. 1, 58–68.

[47] Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics **5** (1943), no. 4, 115–133.

[48] Marek P Michalowski, Selma Sabanovic, and Reid Simmons, *A spatial model of engagement for a social robot*, Advanced Motion Control, 2006. 9th IEEE International Workshop on, IEEE, 2006, pp. 762–767.

[49] Volodymyr Mnih and Geoffrey E Hinton, *Learning to label aerial images from noisy data*, Proceedings of the 29th International conference on machine learning (ICML-12), 2012, pp. 567–574.

[50] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker, *Model-based reinforcement learning: A survey*, 2022.

[51] Ithan Moreira, Javier Rivas, Francisco Cruz, Richard Dazeley, Angel Ayala, and Bruno Fernandes, *Deep reinforcement learning with interactive feedback in a human–robot environment*, Applied Sciences **10** (2020), no. 16, 5574.

[52] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel, *Overcoming exploration in reinforcement learning with demonstrations*, 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 6292–6299.

[53] Amal Nanavati, Christoforos Mavrogiannis, Kevin Weatherwax, Leila Takayama, Maya Cakmak, and Siddhartha S Srinivasa, *Modeling human helpfulness with individual and contextual factors for robot planning*, Robotics: Science and Systems, 2021.

[54] Daniel Neider, Jean-Raphael Gaglione, Ivan Gavran, Ufuk Topcu, Bo Wu, and Zhe Xu, *Advice-guided reinforcement learning in a non-markovian environment*, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 9073–9080.

[55] Andrew Y Ng, Daishi Harada, and Stuart Russell, *Policy invariance under reward transformations: Theory and application to reward shaping*, ICML, vol. 99, 1999, pp. 278–287.

[56] Pierre-Yves Oudeyer et al., *Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner*, Paladyn **3** (2012), no. 3, 136–146.

[57] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek, *Hierarchical reinforcement learning: A comprehensive survey*, ACM Computing Surveys (CSUR) **54** (2021), no. 5, 1–35.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[59] Athanasios S Polydoros and Lazaros Nalpantidis, *Survey of model-based reinforcement learning: Applications on robotics*, Journal of Intelligent & Robotic Systems **86** (2017), no. 2, 153–173.

[60] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar, *A survey on deep learning: Algorithms, techniques, and applications*, ACM Computing Surveys (CSUR) **51** (2018), no. 5, 1–36.

[61] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine, *Learning complex dexterous manipulation with deep reinforcement learning and demonstrations*, arXiv e-prints (2017), arXiv–1709.

[62] Pramila Rani and Nilanjan Sarkar, *Operator engagement detection and robot behavior adaptation in human-robot interaction*, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, IEEE, 2005, pp. 2051–2056.

[63] Charles Rich, Brett Ponsler, Aaron Holroyd, and Candace L Sidner, *Recognizing engagement in human-robot interaction*, Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on, IEEE, 2010, pp. 375–382.

[64] Stephanie Rosenthal, Manuela Veloso, and Anind K Dey, *Acquiring accurate human responses to robots' questions*, International journal of social robotics **4** (2012), no. 2, 117–129.

[65] Stephanie Rosenthal, Manuela Veloso, and Anind K Dey, *Is someone in this office available to help me?*, Journal of Intelligent & Robotic Systems **66** (2012), no. 1, 205–221.

[66] Karina A Roundtree, Jason R Cody, Jennifer Leaf, H Onan Demirel, and Julie A Adams, *Human-collective visualization transparency*, Swarm Intelligence **15** (2021), no. 3, 237–286.

[67] Jyotirmay Sanghvi, Ginevra Castellano, Iolanda Leite, André Pereira, Peter W McOwan, and Ana Paiva, *Automatic analysis of affective postures and body motion to detect engagement with a game companion*, Proceedings of the 6th international conference on Human-robot interaction, ACM, 2011, pp. 305–312.

[68] William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans, *Trial without error: Towards safe reinforcement learning via human intervention*, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2067–2069.

[69] Jürgen Schmidhuber, *Reinforcement learning in markovian and non-markovian environments*, Proceedings of the 3rd International Conference on Neural Information Processing Systems, 1990, pp. 500–506.

[70] Jürgen Schmidhuber, *Formal theory of creativity, fun, and intrinsic motivation (1990–2010)*, IEEE Transactions on Autonomous Mental Development **2** (2010), no. 3, 230–247.

[71] Emmanuel Senft, Séverin Lemaignan, Paul E Baxter, Tony Belpaeme, et al., *Sparc: an efficient way to combine reinforcement learning and supervised autonomy*, Future of Interactive Learning Machines Workshop at NIPS'16, 12 2016.

[72] Isaac Sheidlower, Elaine Schaertl Short, and Allison Moore, *Environment guided interactive reinforcement learning: Learning from binary feedback*

*in high-dimensional robot task environments*, Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (Richland, SC), AAMAS '22, International Foundation for Autonomous Agents and Multiagent Systems, 2022, p. 1726–1728.

[73] Candace L Sidner, Christopher Lee, Cory D Kidd, Neal Lesh, and Charles Rich, *Explorations in engagement for humans and robots*, Artificial Intelligence **166** (2005), no. 1-2, 140–164.

[74] Mohan Sridharan, *Augmented reinforcement learning for interaction with non-expert humans in agent domains*, 2011 10th International Conference on Machine Learning and Applications and Workshops, vol. 1, IEEE, 2011, pp. 424–429.

[75] Kaushik Subramanian, Charles L Isbell Jr, and Andrea L Thomaz, *Exploration from demonstration for interactive reinforcement learning*, Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 447–456.

[76] Richard S Sutton and Andrew G Barto, *Introduction to reinforcement learning*, 1998.

[77] Matthew E Taylor, Nicholas Nissen, Yuan Wang, and Neda Navidi, *Improving reinforcement learning with human assistance: an argument for human subject studies with hippo gym*, Neural Computing and Applications (2021), 1–11.

[78] Andrea L Thomaz, Guy Hoffman, and Cynthia Breazeal, *Reinforcement learning with human teachers: Understanding how people want to teach robots*, ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication, IEEE, 2006, pp. 352–357.

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

[80] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller, *Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards*, arXiv e-prints (2017), arXiv–1707.

[81] Hao-nan Wang, Ning Liu, Yi-yun Zhang, Da-wei Feng, Feng Huang, Dong-sheng Li, and Yi-ming Zhang, *Deep reinforcement learning: a survey*, Frontiers of Information Technology & Electronic Engineering **21** (2020), no. 12, 1726–1744.

[82] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone, *Deep tamer: Interactive agent shaping in high-dimensional state spaces*, Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[83] Christopher J Watkins, *Models of delayed reinforcement learning*, Ph.D. thesis, Ph. D. thesis, Cambridge University, 1989.

[84] Christopher JCH Watkins and Peter Dayan, *Q-learning*, Machine learning **8** (1992), no. 3-4, 279–292.

[85] Thomas Wei, Taylor A Kessler Faulkner, and Andrea L Thomaz, *Extending policy shaping to continuous state spaces (student abstract)*, Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 15919–15920.

[86] Steven D Whitehead and Long-Ji Lin, *Reinforcement learning of non-markov decision processes*, Artificial intelligence **73** (1995), no. 1-2, 271–306.

[87] Baicen Xiao, Qifan Lu, Bhaskar Ramasubramanian, Andrew Clark, Linda Bushnell, and Radha Poovendran, *Fresh: Interactive reward shaping in high-dimensional state spaces using human feedback*, Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020, pp. 1512–1520.

[88] Qianli Xu, Liyuan Li, and Gang Wang, *Designing engagement-aware agents for multiparty conversations*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2013, pp. 2233–2242.

[89] Xin Xu, Lei Zuo, and Zhenhua Huang, *Reinforcement learning algorithms with function approximation: Recent advances and applications*, Information Sciences **261** (2014), 1–31.

[90] Fengji Yi, Wenlong Fu, and Huan Liang, *Model-based reinforcement learning: A survey*, Proceedings of the International Conference on Electronic Business (ICEB), Guilin, China, 2018, pp. 2–6.

[91] Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni, *Robust bayesian inverse reinforcement learning with sparse behavior noise*, Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.