Learning with Humans in the Loop

Gokul Swamy
Anca Dragan, Ed.
Sergey Levine, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-76
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-76.html

May 28, 2020

Copyright © 2020, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Learning with Humans in the Loop

by

Gokul Swamy

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Anca Dragan, Chair
Professor Sergey Levine

Spring 2020

The thesis of Gokul Swamy, titled Learning with Humans in the Loop, is approved:

Chair	[illegible]	Date	5/28/2020
	[illegible]	Date	5/28/2020
		Date	

University of California, Berkeley

Learning with Humans in the Loop

Copyright 2020
by
Gokul Swamy

Abstract

Learning with Humans in the Loop

by

Gokul Swamy

Master of Science in Computer Science

University of California, Berkeley

Professor Anca Dragan, Chair

Robots are starting to leave factory floors and enter daily life, sharing our roads and homes. For this transition to be a smooth one, robots need to be able to learn to interact safely with the highly varied environments and other agents that compose the real world. Much of this task, at some level, requires learning from people, either by (1) developing predictive models of human behavior to be used in motion planning, (2) imitating expert demonstrations produced by people, or (3) learning from user feedback, often given as labels for data. With humans in the loop, learning algorithms should be able to learn *quickly* from data that is both *easy* and *safe* for people to provide. We present methods and experimental evidence that are a useful foundation for developing scalable algorithms for each of these tasks. In Chapter 2, we present experimental evidence for the advantages of a theory-of-mind inductive bias for human motion prediction in the autonomous vehicle domain. In Chapter 3, we propose an algorithm for scaling up assistive teleoperation to multiple robots and validate its efficacy with simulated experiments, a user study, and a hardware demonstration. In Chapter 4, we describe a proof-of-concept system for brain-computer-interface-based shared autonomy via deep learning and present preliminary simulated experiments. In Chapter 5, we provide some conclusions and situate our work in a larger social and ethical context.

To my parents, Shobana Sundaram and Kumar Swamy

Of all the gifts I have been given, being your son is by far the most valuable.

Contents

List of Figures

Acknowledgments

There are so many things I've learned from my advisor Anca Dragan over the years I'm not really sure where to start: from asking truly important questions, to effectively designing experiments to answer them, to making brilliantly clear visualizations of the results. Thank you Anca for teaching me more than I thought there was to know about research and showing me the kind of scientist I want to be one day. Working with you was a dream come true.

The work in this thesis was conducted in collaboration with three amazing graduate student mentors. Dylan Hadfield-Menell's piercing insights and in-depth feedback were crucial to getting much of this work off the ground. Jens Schulz's attention to detail and ability to ask the right questions were critical in refining earlier work. Sid Reddy's almost magical ability to simplify down problems and constant optimism made him a joy to work with.

I'm also incredibly grateful Sergey Levine for his contributions in the form of project ideas, result feedback, and writing suggestions for much of the work in this thesis.

I've benefited tremendously from countless conversations (both research related and otherwise) from members of the InterACT Lab: thank you Andreea, Andrea, Lawrence, Jerry, Rohin, McKane, Hong, Ravi, Sherman, and everyone else for letting me waste so much of your valuable time with complete nonsense. I've been inspired and challenged by students in the RAIL and RLL groups who were excellent karaoke partners and let me sleep on their couches when I didn't want to walk back to my place on southside – a special shoutout to Henry and Daniel.

I could not have made it through the last few years without as wonderful a group of friends I have been lucky to find over the last few years. To name just a few, Andrew and Matthew have been sources of more laughs and emotional support than it is reasonable to expect from anyone. Vasu, Amog, Neel, Ryan, and Rohan have together raised my GPA by more than I'd like to admit and have also been excellent travel and board game buddies. My coworkers from SpaceX (Anthony, Luke, ...) and fellow interns from NVIDIA (Ethan, Jihan, ...) helped me get through some cruel summers. I've had the privilege of being part of various student groups while at Cal but Machine Learning @ Berkeley holds a special place in my heart – thank you for taking a chance on me and showing me I could do this sort of work. Grace and Reo have been great friends for more than a decade now. And despite his best efforts to the contrary, my roommate Arman was unable to keep me from successfully completing any work over the last few years. Jokes aside, I deeply treasure our long impromptu conversations about anything and everything.

I'd also like to thank Mr. Martin Reisert for opening up the world of computers to me.

My parents Shobana Sundaram and Kumar Swamy have been a constant source of support and home-cooked meals throughout my time at Berkeley. Your relentless kindness serves as a constant inspiration and north star for me.

Lastly, I would like to thank all the wonderful restaurants and shops in Berkeley – some of my happiest memories are getting a salad at Sweetgreen, some peach black tea from Asha, a poetry collection from Half Price Books, and reading while I waited for the experiments in this thesis to complete.

Chapter 1

Introduction

All truths wait in all things
They neither hasten their own
delivery nor resist it.

Leaves of Grass
William Wordsworth

1.1 The Necessity of Learning with Humans in the Loop

For applications from autonomous driving to manufacturing, robots are starting to cross into human spaces. The safety and comfort of the people in these spaces should not be thought of in purely abstract terms but as a vital component to ensuring the adoption and positive impact of such systems. Unlike that of inanimate objects, human behavior cannot be predicted purely via physics – people are goal driven, reactive agents that exhibit diverse behavior based on their internal state and environmental context. This makes it a challenge to apply approaches that have been succesful in other areas of robotics and AI. The interplay between randomness and directedness in human behavior makes it appealing for robots to treat them as noisily rational agents that respond to a robot's actions [17] – an approach termed "optimization-centric theory of mind."

However, to determine the form of these optimization-based models of human behavior or to fit less structured predictors, one likely needs to collect data from people. This is because success at a task might be dependent on human convention (e.g. driving through an intersection) or there might be many equally good policies to complete a task and it is hard to know a priori which one a person will choose – approaches like self-play and population-based training are therefore insufficient when applied naively [12].

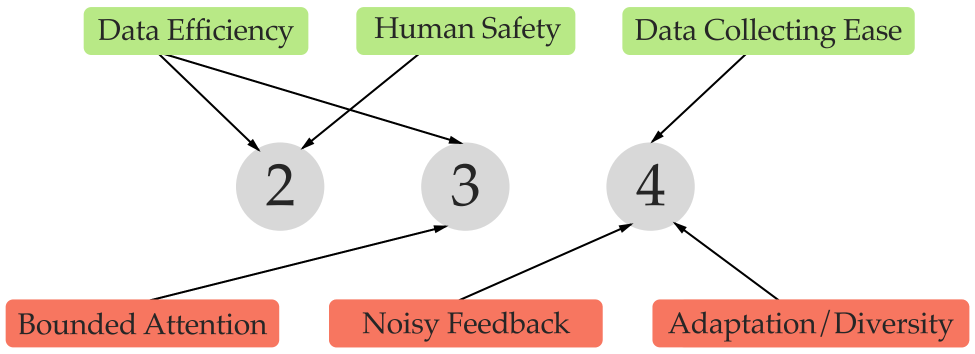

Figure 1.1: A conceptual map of the challenges (red) and goals (green) focused on in each chapter (grey) of this thesis.

1.2 The Challenges of Learning with Humans in the Loop

When we integrate people into the data-collection pipeline of our models, we introduce both challenges and goals into the design of our algorithms.

On the goals side, we want to be able to learn without requiring large amounts of human interaction data. Here, a more structured model of human behavior, while inaccurate in certain circumstances, can drastically reduce the sample complexity of learning algorithms, as we explore in Chapter 2. Limited data collection is desirable even when we're not explicitly modeling human behavior but instead using human-generated labels to learn to perform a task. In Chapter 3, we present an algorithm that naturally collects targeted user demonstrations at areas of particularly high robot suboptimality that could be considered a form of active learning. The ease of collecting a single datapoint is also important to consider when evaluating human-in-the-loop learning systems. Approaches like ours in Chapter 4 that are based on implicit feedback may one day help reduce user data generation burden.

Our results in Chapter 2 also touch on another important desideratum – the safety of the person providing data for the robot. We investigate whether various prototypical algorithms can learn from data of people interacting with each other or more generally from off-policy data with strong constraints on robot behavior.

On the challenges side, we need to deal with the fact that people have inherent limits like bounded attention. In Chapter 3, we present an algorithm to help a single operator supervise many robots by collecting data in situations with few robots where they have enough attention to consider all options and using a cognitive model to generalize to situations with many robots. More generally, users can make mistakes when generating data or we might not be able to completely accurately interpret their feedback. In Chapter 4, we begin to tackle noisy feedback through maximum entropy methods [67].

Another consideration when dealing with data from people is the diversity of human behavior and the fact that even a single person can adapt over time. In Chapter 4, we present a first-pass approach to deal with these concerns via meta-learning [20].

1.3 Terminology

Throughout this thesis, we use *agent* and *robot* interchangeably. We focus mostly on robotics applications but the key ideas are not grounded in the physicality of their instantiations.

Chapter 2

On the Utility of Model Learning in Human-Robot Interaction

He understood that he too was a mere appearance, dreamt by another.

Labyrinths
Jorge Luis Borges

2.1 Acknowledgements

The work in this section was conducted, written up, and shared publicly by the author, Jens Schulz, Dylan Hadfield-Menell and Anca Dragan.

2.2 Introduction

An age-old debate that still animates the halls of computer science, robotics, neuroscience, and psychology departments is that between model-based and model-free (reinforcement) learning. Model-based methods work by building a model of the world – dynamics that tells an agent how the world state will change as a consequence of its actions – and optimizing a cost or reward function under the learned model. In contrast, model-free methods never attempt to explicitly learn how the world works. Instead, the agent learns a policy directly from acting in the world and learning from what works and what does not. Model-free methods are appealing because the agent implicitly learns what it needs to know about the world, and *only* what it needs. On the other hand, model-based methods are appealing because knowing how the world works might enable the agent to *generalize* beyond its experience and possibly explain why a decision is the best one.

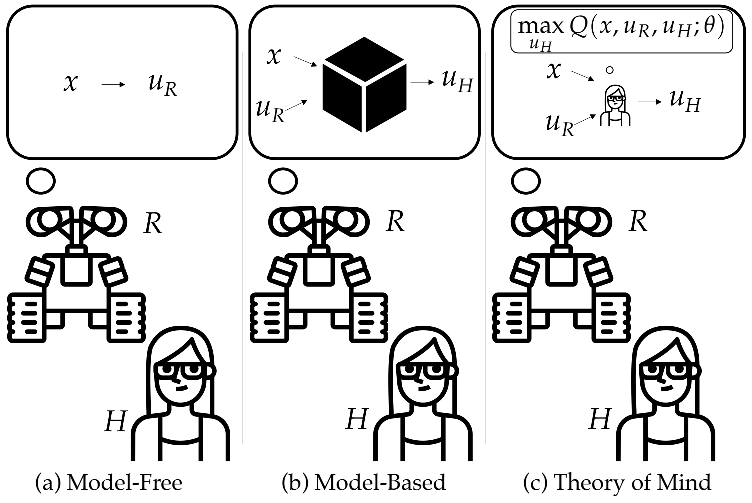

Figure 2.1: We characterize the performance of three HRI paradigms that impose increasingly more structure: (a) in model-free, the robot learns a policy for how to act directly, without modeling the human; (b) in black-box model-based, the robot learns a policy for how the human acts, and uses it when optimizing its reward; (c) in theory of mind, the robot further assumes that the human optimizes a reward function with unknown parameters.

In neuro- and cognitive science, the debate is about which paradigm best describes human learning [23, 37]. On the other side of campus, in AI and robotics, the debate is instead about which paradigm enables an agent to perform its task best. As of today, model-free methods have produced many successes [41, 54, 53], but some efforts are shifting towards model-based methods [21, 47].

In the context of Human-Robot Interaction (HRI), which is our focus in this work, the debate has a different nuance. For robots that do not work in isolation, but in worlds that contain people, the dynamics of the world are no longer just about how physical state changes, but also how *human* state changes – what the human will do next, and how that is influenced by the robot's action. There is thus a lot of richness in what it means to be model-based. On the one hand, the robot can learn a model by observing human state transitions and fitting a policy to them, as it can with any other part of the environment (4.1,b). This is a "black-box" approach to system identification. But on the other hand, the robot can structure its model and explicitly reason about the human differently: humans, unlike objects in the world, have *agency*, and treating them as such means using what Gopnik called a "Theory of Mind" (ToM) [24]: a set of assumptions about how another agent works, including how they decide on their actions (Fig. 4.1,c). Rather than black-box, this is a "gray-box" approach.

When it comes to our own interaction with other people, cognitive science research has amassed evidence that we might use such a theory of mind [13, 22, 59, 7] – in particular, that we assume that others are approximately *rational*, i.e. they tend to make the decisions that are approximately optimal under some objective (or utility, or reward) [8]. But even if humans do use such tools in interaction, it is not at all clear that robots ought to as well. For robots and interaction, methods from all three paradigms exist: model-free [44, 11, 48], regular model-based [52], and ToM-based [68, 18, 6, 30, 51].

Ideally, we'd want to settle the debate for HRI by seeing what works best in practice. Unfortunately, several factors make it very difficult to get a definitive answer: 1) we do not yet have the best ToM assumptions we could get, as research in the psychological sciences and even economics is ongoing; as a result, any answer now is tied to the current models we have, not the ones we could have; 2) making the comparison requires immensely expensive evaluations with robots acting in the real world and failing in their interactions around real people – this is very difficult especially for safety-critical tasks; 3) the answer might depend on the amount of data that can be available to the robot, and the distribution that data is sampled from, which is also hard to predict – therefore, what we need to know is which paradigm to use as a function of the data available.

In this work, rather than attempting to answer the practical and more general question of which paradigm the robot should use, our idea was to turn to a more scientific and focused question. *We do not know how wrong the eventual ToM model will still be, and we do not know how much data we can afford, but we can compare the performance of these paradigms under different possible scenarios.*

We take inspiration from recent work in learning for control [63] that studied the sample complexity of model-based and model-free methods for a very simple system: the linear quadratic regulator. Their idea was that performance in a controlled simple system that we have ground truth for is informative – if a method struggles even on this system, what chance does it have in the real world?

We thus set up a simplified HRI system: we have a robot that needs to optimize a reward function in the presence of a human who optimizes theirs, in response to the robot's actions. We instantiate this in an autonomous driving domain. This simplification from the real world enables us to start answering two fundamental questions: 1) how big the benefit of ToM would actually be, even in the optimistic case of having the perfect set of assumptions about human decision making; and 2) how exactly this benefit decreases as our assumptions become increasingly incorrect.

Findings. We uncover evidence which suggests that if we had a perfect theory of mind, we'd reduce the sample complexity of learning compared to black-box model-based learning, and, more importantly, we might be able to learn solely from off-policy data. In contrast, we found that it is possible for black-box model based approaches to fail to improve even with a lot of data, unless that data comes from the right distribution – the off-policy method failed for us, and even iteratively collecting on-policy data did not seem to be enough in our system without targeted exploration. ToM was surprisingly robust to wrong assumptions: in our driving domain, even when the human deviates from ToM, it could produce useful predictions – but this was in part due to ToM's wrong predictions interacting well with short-horizon planning, rather than ToM making better predictions than black-box approaches. Further, ToM tended to transfer better: when we used a trained model to interact in a new

HRI system, the ToM model performed better even when the human deviated substantially from training behavior.

Limitations. These findings should be of course taken with a grain of salt. The differences we introduce between the ToM and the ground truth ”human” might not be representative of what differences we will see in real life. We also do this for one particular task, with one particular ToM instantiation. Because of the difference between accurate predictions and *useful* predictions, different tasks might lead to different results. In a previous iteration of this work, we investigated a task where a robot was attempting to influence the human driver from behind to move out of the way: we realized that most wrong predictions would lead to the correct plan in such a scenario, because most wrong predictions would lead to the human moving out of the robot's lane. We specifically designed the current task so that being able to capture the influence on the human would actually matter more to task performance. This is not true of all tasks. Further, the realities of planning in continuous state-action spaces even in our relatively simple scenario, such as limited planning horizons and convergence to local optima, put their own stamp on the findings. That said, the results give us a glimpse at what might happen, pointing to surprisingly nuanced trade-offs between ToM and black-box approaches, in particular with respect to the type of data required and the ease of initialization.

Overall, we are excited to contribute a quantitative comparison of these paradigms for an HRI task, along with an analysis of their degradation as we make the wrong modeling assumptions, decrease the amount of data available, or restrict the ability to collect data on-policy and to explore.

2.3 Interaction Learning Algorithms

Notation

For all of the following methods and experiments, we denote the human plan at time t by $\mathbf{u}_H^t$, and the robot plan at time t by $\mathbf{u}_R^t$. We also denote the action executed by the human at time t by u_H^t, and for the robot u_R^t. Both the human and robot states at time t are denoted by x_H and x_R.

Robot Objective

The robot's goal is to optimize a reward function $r_R(x_R, x_H, u_R, u_H)$ that depends on both its state and action, as well as the human's state and action.

Theory-of-Mind-Based Learning

In line with previous work [22, 59, 7], our ToM will assume that the human optimizes a reward function. The robot will focus the learning on figuring out this reward via inverse reinforcement learning, and the ToM-based method will plan the robot's actions using the learned model.

In particular, the reward will have the form

$$r_H(x_H, x_R, u_H, u_R; \theta) = \theta^T \phi(x_H, x_R, u_H, u_R)$$

where θ is a vector of weights and ϕ is a feature map from the current state of the system, which depends on the robot's state and action as well, akin to the robot's reward. We describe the particular features we assumed in more detail in Sec. 2.4.

To optimize their reward, the person needs to reason about the inter-dependency between their future actions and those of the robot. Work on ToM has investigated different ways to capture this, from infinite regress (i.e. "I think about you thinking about me thinking about you...") to capping the regress to one or two levels. Our particular instance of ToM is based on prior work which avoids regress by giving the human access to the robot's future plan [51], and is given by

$$\mathbf{u}_H^*(\mathbf{u}_R; \theta) = \arg\max_{\mathbf{u}_H} \mathcal{R}_H(x_H, x_R, \mathbf{u}_H, \mathbf{u}_R; \theta) \tag{2.1}$$

with $\mathcal{R}$ denoting the cumulative reward, i.e. the sum of rewards over a finite horizon of the length of the trajectories $\mathbf{u}_H$ and $\mathbf{u}_R$. This assumption is very strong, i.e. that the human can read the robot's mind. Part of our goal is to simulate the human as instantiating other approaches, and teasing out how useful or not ToM still is when its assumptions are wrong. This particular ToM instance will also assume that the person computes this at every step, takes the first action, observes the robot's new plan, and replans.

To leverage this ToM model, the robot needs to know the human reward function r_H. In our experiments, we create a dataset of demonstrations $\mathcal{D}$ by placing our ground-truth human (be it a human perfectly matching the ToM model or one that does not) around another vehicle executing various trajectories, and recording the human's response. In the real world, this data would be collected from people driving in response to other people. We then run inverse reinforcement learning [1, 38, 67, 51] on $\mathcal{D}$ to obtain weights θ.

Finally, given the human reward function described by the learned weights θ, the robot optimizes its own plan:

$$\mathbf{u}_R^*(\theta) = \arg\max_{\mathbf{u}_R} \mathcal{R}_R(x_R, x_H, \mathbf{u}_R, \mathbf{u}_H^*(\mathbf{u}_R; \theta))$$

with $\mathcal{R}_R$ denoting the robot's cumulative reward. The robot plans, takes the first action, observes the next human action, and replans at every step. We use a Quasi-Newton optimization method [4] and implicit differentiation to solve this optimization problem.

The main challenge with the ToM approach is that in reality people will not act according to $\mathbf{u}_H^*(\theta)$ for *any* θ. In general, ToM models will be misspecified, failing to perfectly capture human behavior.

Black-Box Model-Based Learning

The Theory-of-Mind approach models the human's actions as explicitly optimizing some cost function. An alternative is to learn this function directly from data via, e.g., a conditional neural network, as in [52].

Off-Policy (MB-OFF). In the "off-policy" black-box model-based approach, we collect a training dataset $\mathcal{D}$ of human-robot interactions as in the ToM approach and fit a neural network f to $\mathcal{D}$. This allows us to estimate the human plan $\mathbf{u}_H$, given the human and robot histories, the current state of the system, and the robot plan:

$$\mathbf{u}_H = f(H_R, H_H, x_R, x_H, \mathbf{u}_R).$$

H_R and H_H are the state-action histories of the robot and human over a finite time horizon. The model f is trained by minimizing the loss between $f(.)$ and the observed data $\mathbf{u}_H$ in $\mathcal{D}$. Specifically, we use a neural network with three fully connected layers, each with 128 neurons and ReLU activations, and train the network using ADAM [33].

Given this learned model, the vanilla model-based method generates a plan with trajectory optimization, just like the ToM method:

$$\mathbf{u}_R^*(f) = \arg\max_{\mathbf{u}_R} \mathcal{R}_R(x_R, x_H, \mathbf{u}_R, f(H_R, H_H, x_R, x_H, \mathbf{u}_R)).$$

The distinction is whether the prediction about $\mathbf{u}_H$ comes from optimizing $\mathcal{R}_H$, or from the black box predictor f.

On-Policy (MB-ON). The off-policy approach ignores the fact that the predictive model influences the behavior of the robot and therefore the future states on which the robot will query the predictor. In fact, if f attains a test error rate of ϵ it can still exhibit prediction errors that are $O(\epsilon T^2)$ when rolled out over a horizon T [50]. The trajectories generated by optimizing against a fixed model induce *covariate shift* that reduces the accuracy of the model.

To deal with this, a typical approach in system identification is to alternate between fitting a model, and using it to act and collect more interaction data. This comes at a cost: the need for data collected on-policy, from interacting with the human, rather than from (off-policy/offline) demonstrations.

In this method, we again use a neural network f to predict the human actions. But unlike the previous model-based method, which relies on training data collected beforehand, here we train f *iteratively.*

Of course, needing interaction with the human can be prohibitive, especially if a) a lot of interaction data is needed, or b) the robot does not perform well initially, when its model is not yet good, which can harm adoption or lead to safety concerns.

Idealized (MB-I). Even collecting data on-policy has no guarantee that the robot will acquire a human model that is conducive to the *optimal* robot policy, i.e. if we assume the human acts according to a function f_{GT}, the robot learning on-policy might not converge to $\mathbf{u}_R^*(f_{GT})$. This is because the robot might not explore enough. However, getting exploration right is an open area of research. We thus also test a method that uses idealized exploration – we collect data not only on-policy, but rather also "on-the-optimal-policy". This method in a sense "cheats" by getting access to data collected by acting according to $\mathbf{u}_R^*(f_{GT})$ from different initial states. We then mix this data with on-policy data, iteratively, to obtain a model that can cover both the current and previous distributions, as well as the idealized distribution (we found that this mixing was important to the performance compared to using the idealized distribution alone). The reason this method is worth considering, despite the very strong assumption of access to the idealized distribution for "exploration", is that robots might be able to use human-human interaction data in situations where that optimal robot policy might coincide with what people do with each other.

Model-Free Learning

A final approach is to employ fully model free methods, such as policy gradients or DQNs. These methods are quite general and fast at inference time as they make no assumptions about the environment and don't explicitly plan online. We use Proximal Policy Optimization (PPO) [53], a model free reinforcement learning algorithm that has had strong results in other continuous control tasks. Although more sample efficient approaches exist, we selected PPO because it has been adopted as a sort of baseline for continuous control tasks.

PPO works by computing clipped gradients of expected reward with respect to the parameters of a policy. This gradient is estimated with rollouts using the current policy parameters in the environment. The algorithm alternates between rolling out trajectories and performing gradient updates. See [53] for a more complete explanation of the approach. We used the PPO2 implementation in the Stable Baselines repository [26] with default parameters as our model-free algorithm. The primary difficulty in applying PPO to this setting is that the human simulator we implemented (for testing what happens where ToM assumptions are exactly right), does not, strictly speaking, fit into the environment model used for reinforcement learning. Because the human reacts to what the robot *plans* to do in the future, the environment is different depending on the current policy parameters. We implement this by having the human respond to a robot plan that was generated assuming the person continues at a constant velocity for the rest of the planning horizon. Because of how precise the controls need to be for this task, we do not sample from the distribution over actions the robot's policy outputs during evaluation time and instead use the mean. We also clip some of the penalties for catastrophic events like driving off the road or colliding with another vehicle and terminate the episode early to not poison the reward buffer of the policy gradient algorithm.

2.4 Performance under Correct Modeling Assumptions

We begin with comparing these methods in a driving domain.

Experiment Design

(Ground Truth) Human Simulator. For our driving domain, states are tuples of the form (x, y, v, α), where x and y are the positional coordinates, v the speed, and α the heading. The actions are $u = (a, \omega)$, where a is a linear acceleration, and ω an angular velocity.

The ground truth human simulator plans forward over a finite time horizon T (in all experiments, $T = 5$) by optimizing over a linear combination of features:
Car Proximity: This cost is based on the distance between the robot and human cars, which represents human's desire to not hit another vehicle. Given the human state $(x_H, y_H, v_H, \alpha_H)$ and the robot state $(x_R, y_R, v_R, \alpha_R)$, this cost is given by $\mathcal{N}((x_R, y_R)|(x_H, y_H, \sigma^2_{car})$.
Lane Edge Proximity: This cost is based on the distance to the nearest lane edge. This represents how humans generally prefer to stay in the middle of their lane. Letting the left edge of some lane be L_l and the right edge L_r, the lane cost is given by: $\mathcal{N}(L_1|x_t, \sigma^2_{lane}) + \mathcal{N}(L_r|x_t, \sigma^2_{lane})$.
Forward Progress: This cost is based on the vertical distance between the next state and the current state, representing how humans want to go forward when driving. This cost is given by: $-(y_{t+1} - y_t)$.
Bounded Control: This cost is based on accelerating or trying to turn more quickly than certain bounds, representing how humans prefer smoother rides and cars have actuator limits. The cost is given by $\exp(a - a_{max}) + \exp(\omega - \omega_{max})$.
Offroad: This cost represents how drivers want to stay on the road when driving. Letting the left edge of the road being R_l and the right edge R_r, the offroad cost is given by $\exp(x_t - R_l) + \exp(R_r - x_t)$.

The weights on these features were tuned to produce plausible/natural driving in a series of scenarios. In addition to the features and weights, the ground truth human simulator is given the plan of the robot $\mathbf{u}_R$. It then solves the cost minimization in (2.1). Importantly, this particular simulator exactly matches the assumptions made by the ToM learner (Section 2.3), in both the features used and planning method. The next section modifies the simulator so that the ToM assumptions are wrong.

Environment. The experiment environment consists of the human and robot car on a three-lane road, with the robot beginning in front of the person. The robot car starts in the middle lane while the human car starts in either the middle or left lane. The robot has a similar set of features to that of the human, incentivizing it to make progress, avoid collisions, keep off of the lane boundaries, and stay on the road. However, we add another feature representing the penalty associated with colliding into a pair of trucks that are in front of the robot and occupy the center and right lanes.

We chose this as our environment because for all its simplicity, it can actually capture sophisticated interaction: given our ground truth human, to do well, the robot should merge in front of the person in the left lane by *influencing the person to make space* rather than passively waiting for the person to pass by and accumulating significantly less reward for forward progress. We use this scenario as an interesting challenge for HRI, because it requires being able to account for robot's *influence* on the human. Accordingly, we initialize the ToM

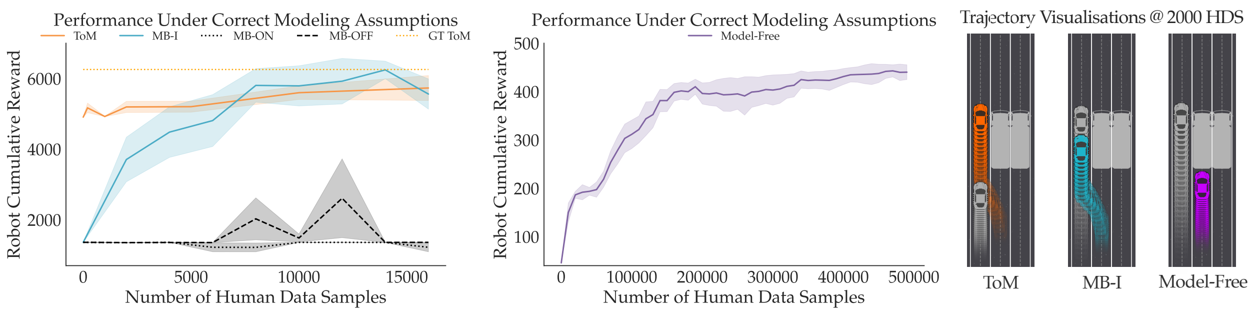

Figure 2.2: The test rewards of the interaction learning algorithms on the scenario with the ground truth human simulator. The ToM learner has the smallest sample complexity, followed by the MB-I method. The MB-ON and MB-OFF methods do not appear to improve from their original performance. The ToM is able to consistently merge in front of the human car most quickly, and thus obtains the highest reward, while on average, the MB-I method waits for the person to pass before merging.

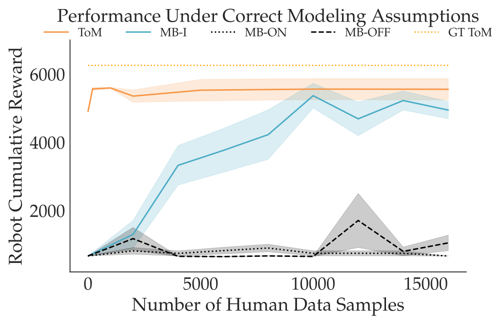

Figure 2.3: The comparison between ToM and black-box model-based for modification of our environment setup where there are no control bounds for the cars. This matches the result from Fig. 2.2

and black-box model-based methods by pre-training them on a dataset in which the human drives in isolation – this makes it so that both methods get access from the start to the basics of human driving (staying on the road, in the lane, accelerating to reach the speed limit), without giving away any of the aspects that are important to this particular task, i.e. the influence.

Manipulated Variables. We manipulate two variables: the interaction learning algorithm (with the options described in the previous section) and the amount of data (number of samples) the learner gets access to.[1] For the off-policy methods, we use data from a different

[1]Note that the u_R and u_H collected in training data and used in learning algorithms are not *plans*, but rather *actions* that have been executed by the robot and human, since in reality, humans can only react to

type of interaction where the robot drives at constant velocity near the human.

Dependent Measures. In each experiment, we measure the reward the robot accumulates after training over test environments drawn from a more challenging subset of the training environment initial state distribution. We train ToM and model-based 5 times with each data sample size and measure test reward for each trained model. We train model-free only four times due to the several orders of magnitude larger amount of data required.

Analysis

We plot the results and visualize trajectories from each paradigm in Fig. 2.2. We see that ToM starts with higher performance even at 0 samples. This is because while both models are pre-trained with a data from a human acting in isolation and thus make very similar predictions, the *gradients* of those predictions with respect to the robot's plan are different, and that in turn leads to different plans. With the black-box model, the gradients are close to 0 due to the non-reactive training data. This results in the robot planning to let the human pass and merge behind. The ToM model, on the other hand, anticipates influence on the human's plan because of its structure (i.e. the collision avoidance feature), despite pre-training on non-reactive data (which settles on a nonzero weight for said feature). This enables the robot to figure out that it can merge in front of the human even from the start. This speaks to a certain extent to the ease of initializing ToM: because it encodes that people want to avoid collisions, and that human actions are influenced by robot actions, the original solution, albeit imperfect, is already qualitatively achieving the desired outcome.

As the ToM model gets off-policy data from the ground truth human simulator, the robot's reward when planning with ToM improves. Looking at a representative trajectory (right of Fig. 2.2), we see that the robot gets in front of the human, who slows down to accommodate.

On the other hand, the off-policy model-based method, which learns from the same demonstrations used by the ToM approach, is unable to learn to merge in front of the human car. One explanation for this behavior is the difference between the training and test distributions – at test time the robot plans with the learned model, generating different kinds of trajectories with different human responses than in training. Concretely, the off-policy data has a mild influence of the robot's actions on the human, whereas solving our test merge task requires being able to model stronger influence. The ToM model can better cope with this because it has access to the strong predefined structure various features in the reward function, and can rely on trajectory optimization to produce the corresponding human trajectories in new situations, given new robot trajectories.

What is more surprising at a first glance, however, is that the on-policy method also fails to improve. This points to a potentially fundamental problem with model-based methods in this type of interactive situation: they start not knowing how to influence the human, so the initial policy does not attempt to merge in front and instead lets the human pass.

and learn from actions they can physically observe.

Then, they collect data on-policy, but none of the roll-outs end up influencing the human's behavior enough, so their model never evolves to capture that potential stronger influence. As a result, the robot's policy never attempts the merge and remains at a low reward.

The idealized model-based, which "explores" by getting access to data from the optimal policy in response to the *ground truth* human model, does get up to the ToM performance, but requires more data. We also see these results reproduced in a setting where we eliminated the control bounds on the cars (Fig. 2.3).

For model-free learning, we used the PPO2 implementation in the Stable Baselines repository [26] with default parameters as our model-free algorithm. This method takes several orders of magnitude more than the model-based methods and is not able to match their performance. This can likely be attributed to the fact that the model-free method is not handed the dynamics of the system and therefore cannot take advantage of online planning, which is fundamental to the way model-free methods operate. Additionally, the accuracy of the controls required to successfully perform a merge make exploration challenging in this scenario, so the Gaussian noise used by PPO2 might also play a factor in the algorithm's relatively poor performance.

Takeaways. Overall, what we find confirms intuition: if we have a good model, learning its parameters leads to good performance compared to learning from scratch. More surprising is the performance of the black-box methods, both off- and on-policy. Without seeing data of the human responding to the robot actually merging in front of them (which is what the idealized exploration brings), the black-box approach did not seem to be able to model this, and got stuck in the mode where the optimal solution is to go behind the human. In contrast, ToM could more easily make that leap from the data it was trained on. In our case, the pre-training already brought it most of the way there; in general though we expect that ToM can for instance take off-policy data of mild influence (when the robot is further away) and extrapolate it to stronger influence (when the robot is closer to the human) – this seems to be difficult for the black-box model, judging by the off-policy black-box results. On the other hand, had we initialized ToM differently, and had the off-policy data not contain any influence, ToM would have looked the same as the on- and off-policy black box methods, and it is entirely possible that not even an on-policy ToM could have fixed that. Overall, the results speak mostly to the difficulty that black-box models might have in extrapolating from one pattern of interaction to another, because they lack that structure that ToM exploits.

2.5 Performance under Incorrect Modeling Assumptions

From what we have found in the previous section, Theory of Mind is appealing because it has low sample complexity and works from off-policy data. However, the bias introduced with this approach could lead to underfitting. To quantify this, we compare the ToM and model-based methods when we modify the human simulator. Because of the tremendous

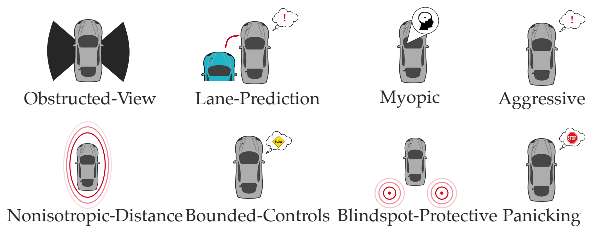

Figure 2.4: The various modifications made to the ground truth human simulator. The first row corresponds to modification of planning methodology, while the first three elements of the second row correspond to changes in reward features. The last corresponds to an irrational planning heuristic humans might use while under pressure.

difference in terms of performance and sample complexity of the model-free method, we chose to omit it to focus on the aforementioned comparison.

Human Simulators that Contradict Modeling Assumptions

Inconsistency in how humans plan, the "features" they might care about, or unexpected reactions to certain actions all violate the assumptions made by the ToM learner. We aimed to create ground truth modifications that are analogous to differences between reality and our ToM-based modeling – things that designers of these systems might get wrong. As such, even though these are controlled experiments where we know the ground-truth, we think they provide some indication of how real-world differences would look like under different hypotheses. We group these modifications into 3 categories:

Incorrect model of how the human plans

One way our model could be wrong is if it inaccurately captures the planning process – even if we assume the person is actually trying to optimize for a known reward, they might not optimize well or reason about the robot differently than ToM assumes.
"Obstructed-View" – Humans have blind spots. Our instance of ToM assumes humans have a 360° vision. In reality, drivers have blind spots. To model this we hide cars that are not in a double-cone from the human, with a vertex angle of 45°. Robots that do not model blind spots will thus take more risky maneuvers, expecting to be seen by the person when they are not.
"Lane-Prediction" – Humans can plan conservatively. Given the inherent risks in driving, humans may be more cautious in their planning than necessary, taking evasive maneuvers when there is any chance of danger. This simulator swerves out of the current lane if the robot angles itself slightly towards said lane. A robot that does not know this might not be able to influence the human as much is possible. Note that this no longer matches our ToM's assumption that the human gets access to the robot's plan.

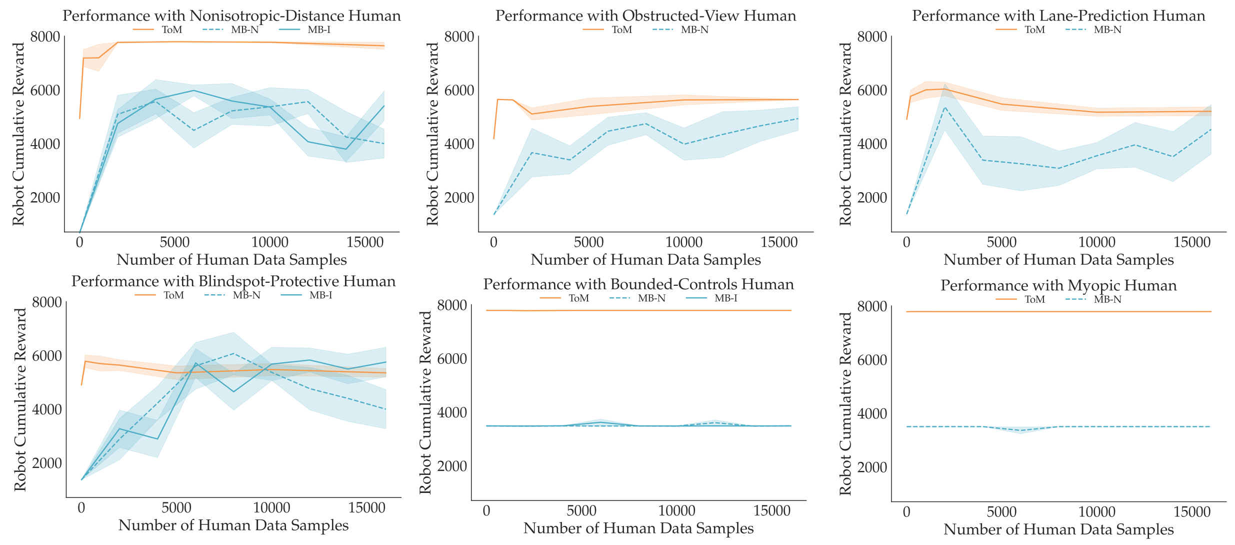

Figure 2.5: ToM vs black-box model-based on different simulators. ToM is relatively robust to simulator modifications. This is in part due to the structure it is imposing being useful in low sample complexity regimes despite being wrong, but also because its predictions end up being wrong in a way that help compensate for the robot's planning suboptimality (see Fig. 2.6).

"Myopic" – Humans might not plan ahead for as long as the robot assumes. Another assumption ToM makes about the human's reasoning is that it plans as far forward as the robot does. In reality, however, humans may be more myopic, and plan forward for a shorter time horizon than we assume. Our "Myopic" human simulator only plans forward for one step.

Incorrect model of what the human cares about

Another class of inconsistency in modeling deals with reward features.

"Nonisotropic-Distance" – Humans care about avoiding other cars, but we might not know how sensitive they are to getting close to different areas of another car. The original human simulator has a cost based on a Gaussian that takes in the Euclidean distance between the centers of the two cars, the "Nonisotropic-Distance" simulator modifies the cost contours to be longer than they are wide, as show in Figure 2.4. This models the fact that some people are more comfortable with cars to their sides rather than in their own lane.

"Bounded-Controls"– We might not know people's preferences for speed or their control limitations. Human drivers have different preferences for how fast they turn or accelerate as well as cars with different control bounds. To model this, the ground truth

human simulator's values of a_{max} and ω_{max} (see section 3.A) are reduced to $\frac{a_{max}}{2}$, $\frac{\omega_{max}}{2}$, reducing the capability of the human to react to the robot.
"Blindspot-Protective" – Humans might additionally care about not having another car in their blindspot. A subclass of modifications we have not yet considered is where the human might use features in planning that the robot might not know about. One example of this might be discomfort with having cars in one's blindspot. Drivers might speed up or slow down to avoid such an arrangement. This modification models this dislike by adding additional points of Gaussian cost where blindspots are, as illustrated in Fig. 2.4.
"Panicking" – Humans might be heuristic-driven. Humans might also use heuristics that are irrational. Our "Panicking" modification combines the slower speed of the "Bounded-Controls" modification with an additional heuristic of stopping immediately if another car is fewer than 2 car-lengths behind. This is inspired by a newer driver, whose inexperience causes him to drive slowly and panic when another car approaches.

A modification purposefully detrimental to the robot

Our last modification is meant to purposefully create dangerous situations when the robot is using the ToM assumption. **"Aggressive" – Humans might want to stay in front of autonomous cars.** New autonomous vehicles might seem dangerous to drivers and they may want to stay in front of them so they do not have to worry about false stops by the car. This modification is implemented by flipping the sign of the car-avoidance feature and moving the center of the Gaussian to a spot in front of the robot car.

Analysis

Fig. 2.5 and Fig. 2.6 plot the performance of ToM and black-box model-based[2]. For the latter, we used the idealized distribution when possible by getting the robot to plan with the ground truth modified model and collecting data. Additionally, we also collected data from the robot planning with the unmodified model (”MB-N" in the plots, which performs comparably).

Overall, we see the ToM performing better initially across the board, and black-box catching up in most modifications with enough data. Surprisingly, we do not see black-box surpassing the performance of ToM. We found this to be caused by an interesting facet of planning in these environments, rather than by the predictions failing to improve. ToM also ends up with imperfect predictions, but the robot planning with it achieves higher reward nonetheless. This is because our planner uses a short horizon and is locally optimal. As the trajectories in Fig. 2.6 show, the ToM method incorrectly predicts that the aggressive human will slow down more than they actually do. This fools the short horizon planner into deeming that merging in front of the human is the right option. While this is not actually right with respect to reward over that short horizon when interacting with the aggressive human, it

[2]Panicking results were analogous to bounded-controls and myopic and were ommitted from the plot.

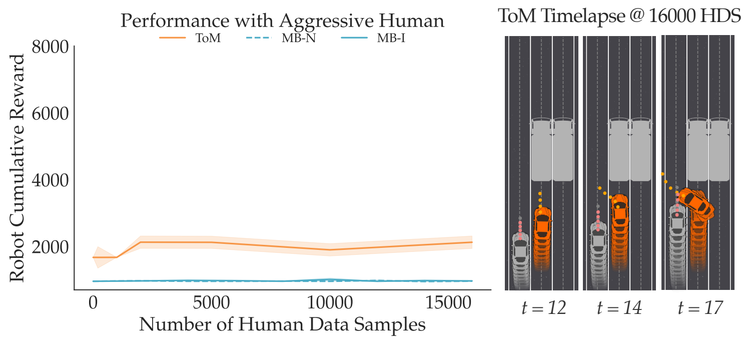

Figure 2.6: Rewards and sample trajectories for the aggressive human experiment. We see ToM doing better whereas black-box model based fails to improve the robot's reward as it gets more data. This is not so much a reflection on the predictions, but rather on the influence these predictions have on the planning output: ToM wrongly predicts the person would slow down so it deems optimal to merge in front when using a short horizon planner; this is not actually optimal for the short planing horizon when interacting with the "real" human, but leads to higher cumulative reward over the entire horizon.

ends up being the right decision for the long term, at least according to the reward function we specified. The robot ends up with an aggressive merge which accumulates more reward overall than the black-box model-based method. In a few of the modifications, the black-box approach failed to improve the robot's performance with more data, again because of the difference between prediction accuracy and planning with those predictions when using a suboptimal planner. There were also cases where ToM obtained maximum reward from the start, and that is because some modifications make the human slower and less aggressive, which enables the merge to happen seamlessly even from the start, when ToM operates solely based on the pre-training.

Takeaways. The ToM performance tends to quickly asymptote, and it typically does so at a higher reward that the black-box methods. This is in part because these misspecified ToM models make wrong predictions that help compensate, in our environment, for the planner's short horizon. While we expect that in general, when given data from the right distribution, black-box model-based asymptotes to higher robot reward than ToM, our finding speaks to the intricate relationship between prediction and planning, and how better predictions do not always lead to better plans. This is a subtle aspect that practitioners will need to consider in the design of algorithms for robots that interact with people.

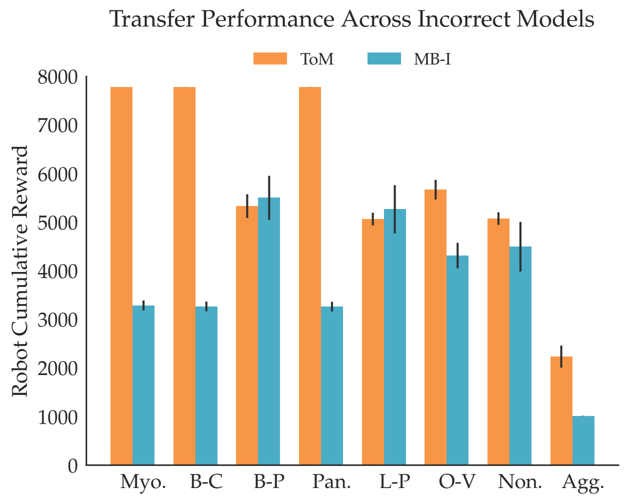

Figure 2.7: Results from testing the models trained with the original human simulation in the modified simulations. Overall, ToM tends to transfer better.

2.6 Transferability of Learned Models

Finally, we study the transferability of the models trained against the original human. We compare their performance when tested against the modified simulators from the previous section. Fig. 2.7 shows that all models transfer better when changes are small.

Some modifications (myopic, bounded controls, and pankicking) make the human go slower in our environment, thereby making the merge that ToM attempts have high reward (even higher than with the original human). Despite the change in human behavior, ToM keeps making predictions that enable the robot to merge. On the other hand, the shift in the input distribution causes poor predictions from black-box model based, which in turn leads to poor performance. ToM also handles the aggressive and obstructed-view humans better. The blindspot-protective and lane-prediction humans behave similarly enough to the original human that both methods retain their original performance.

2.7 Discussion

Summary. We provided what is, to the best of our knowledge, the first comparison between model-free, black-box model-based, and Theory-of-Mind-based methods for interaction. We used a simple driving task to quantify the performance advantage of ToM-based when we have made the right assumptions, as well its data collection advantage: it does not seem to require on-policy interaction data during learning, and can be trained on observed human-human data.

We also found that black-box model-based methods can perform as well as ToM, but the training distribution matters: in our example, neither off- nor on-policy was successful, and instead the robot added exploration to on-policy via an idealized distribution (coming from the ground truth optimal policy). These methods did not seem able to extrapolate from mild effects of the robot on the human (in off-policy data) to strong effects (needed in the test environment). They also required much more data (an order of magnitude in our experiments). Further, we found that the model-free method required several orders of magnitude more data and was not able to achieve anywhere near the performance of the other methods considered.

We also studied what happens as ToM's assumptions become increasingly wrong by emulating the kinds of deviations we might expect to encounter. Lastly, we saw that ToM methods seem to transfer better.

Limitations. Ultimately, our work does not answer the question of which type of model to use, it merely provides a few data points that are based on hypothetical human behavior. Our results are also for a particular domain, environment, and a particular ToM model. There are a few major caveats to these results as a consequence. First, the interaction with the planner suboptimality led to ToM actually benefit from its wrong assumptions. In other enviroments, the opposite might be true, i.e. that ToM might combine particularly poorly with a short planning horizon. On average, we expect that black-box model-based methods eventually surpass the ToM performance with enough data, unlike what happened in our environment. Second, even with perfect planners, more accurate prediction does not always mean better performance, which can again affect these results. In some environments, such as the robot starting behind the person and accelerating to move the person out of the way, random predictions are just as useful as accurate ones, and a black-box approach would perform really well from the start. Third, the environment also made it so that ToM could start with a very high performance just based on pre-training. It is not clear how much easier it is to pre-train ToM generally. Further, even though on-policy black-box model-based did not perform well in our environment, this can be addressed with better exploration without requiring the idealized distribution (at the cost of data and cumulative regret), or by mixing in relevant enough human-human interaction data.

Overall, these results only scratch the surface of understanding the trade-offs with using Theory-of-Mind approaches for inductive bias in HRI. These trade-offs are not just based on sample complexity, but are instead complicated by the interaction between prediction accuracy and planning suboptimality, and by the feasibility of collecting on-policy data and of exploring. Looking forward, we are also excited to research *hybrid* methods that might be able to take advantage of the best of each paradigm: using ToM assumptions in low-data regimes, having flexibility when there is enough data.

Chapter 3

Scaled Autonomy: Enabling Human Operators to Control Robot Fleets

A certain type of perfection can only be realized through a limitless accumulation of the imperfect.

Kafka on the Shore
Haruki Murakami

3.1 Acknowledgements

The work in this section was conducted, written up, and published by the author, Siddharth Reddy, Sergey Levine, and Anca Dragan.

3.2 Introduction

Sliding autonomy [16, 34, 10, 56] is a promising approach to deploying robots with imperfect control policies: while a fleet of autonomous robots acts in separate environments, a human operator monitors their states, and can intervene to help a robot via teleoperation when the robot encounters a challenging state. Imagine, for instance, a fleet of delivery robots: at any given time, most of them may be driving in easy conditions with few pedestrians, while one of them encounters a crowded sidewalk and requires operator intervention. Ideally, the performance of such a human-robot centaur team would scale smoothly with the increasing capabilities of the robot: as autonomy improves, challenging states become rarer, and a single human operator should be able to control a larger fleet of robots.

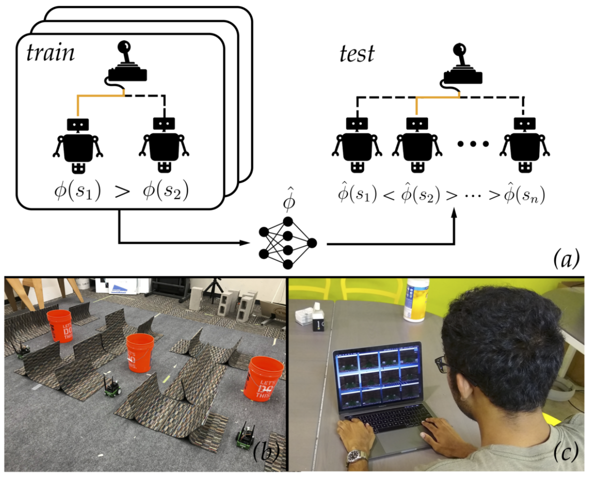
train
$\phi(s_1) > \phi(s_2)$
$\hat{\phi}$
test
$\hat{\phi}(s_1) < \hat{\phi}(s_2) > \cdots > \hat{\phi}(s_n)$
(a)
(b)
(c)

Figure 3.1: We learn which robot a user would prefer to control, by observing the user manage a small number of robots. We then use the learned preference model to help the user control a much larger number of robots. We evaluate our method (a) on a simulated navigation task and real hardware demonstration (b) through controlled, synthetic experiments with expert agents that stand in for users, and a human user study with twelve participants (c).

Unfortunately, while the user may be a skilled operator, they have limited attention. As the fleet grows larger, the user's ability to maintain awareness of the states of all robots and identify the robot that most requires intervention degrades.

We propose to overcome this challenge by automating the operator's choice of which robot to control. Given a large number of robots running in separate environments, our approach is to train a model that predicts which robot the user would teleoperate, if the user had the ability to analyze all the robots' current states quickly enough. Our insight is that we can use decisions that the operator makes in easy settings with only a few robots, where they can feasibly pay attention to all the robots' current states, to train a predictive model of user behavior that generalizes to challenging settings with many robots.

The key to generalizing the user's choices from easy to hard settings is to treat them as observations of relative preferences between robots: every choice the person makes to control one particular robot instead of any of the other robots is assumed to be an approximately optimal decision, with respect to maximizing the user's utility function. Every choice gives us information about that utility, namely that the utility of controlling the chosen robot was higher than the utility of controlling any other robot. We can thus use observations of the operator's choices to fit a model of their utility function. At test time, we apply the learned model to the current state of each robot, and automatically switch the user to controlling the robot with the highest predicted likelihood of being chosen.

We test our method in simulation and through an in-person user study, on a simulated navigation task and real hardware demonstration. In the navigation task, the robot must successfully navigate through a video game environment with hazards and health packs to reach a goal state (see schematic in Figure 3.4). We initially evaluate our method in synthetic experiments where we simulate user input under ideal assumptions, and where we have access to ground-truth user preferences. We find that our method effectively generalizes the user's choices in easy settings with a small number of robots to challenging settings with a large number of robots. We also find that modeling the user's choices as a function of relative preferences between robots is important for this generalization. To show that our results extend to real user data, we conduct an in-person user study with twelve human participants, where we evaluate each participant's ability to manage twelve robots with and without assisted choice. We find that assisted choice enables users to perform significantly better than they can on their own.

3.3 Related Work

In shared autonomy, a human operator and robot collaborate to control a system which neither the operator nor the robot could control effectively by themselves [2]. Previous work in this area [30, 19, 48, 55, 9] has focused on some combination of inferring user intent and acting to achieve it. We instead focus on helping the user process information quickly enough to manage a fleet of robots. The problem we tackle is more akin to that addressed by a continuously-running search engine like the Remembrance Agent [49], which assists a user's decision-making by displaying information relevant to the user's current context.

The closest prior work is in sliding autonomy [16, 34, 10, 56] and active learning [15], where the robot can request user intervention in challenging or uncertain situations. Prior methods tend to require knowledge of the task in order to determine when user intervention is needed. Our method makes minimal assumptions about the task, and instead allocates the user's attention using a learned model of the user's preferences. To our knowledge, we are the first to use a general-purpose learning approach to allocating operator interventions.

3.4 Learning to Allocate Operator Interventions

Our goal is to help the user choose which robot to control at any given time. To do so, we learn to mimic the way the user manages a small number of robots, then use the learned model to assist the user in controlling a large number of robots.

Problem Formulation

We formalize the problem of automating the operator's decision of which robot to control as one of estimating the operator's *internal scoring function*: a function that maps the state of a robot to a real-valued score of how useful it would be to take control of the robot, in terms of maximizing cumulative task performance across robots.

User choice model. Let $[n]$ denote $\{1, 2, ..., n\}$, $i \in [n]$ denote the i-th robot in a fleet of n robots, and $i_H^t \in [n]$ denote the robot controlled by the user at time t. We assume the user selects robot i_H^t using the Luce choice model [40],

$$\mathbb{P}[i_H^t = i] = e^{\phi(s_i^t)} / \sum_{j=1}^{n} e^{\phi(s_j^t)}, \tag{3.1}$$

where $\phi : \mathcal{S} \to \mathbb{R}$ is the user's scoring function, and s_i^t is the state of robot i at time t. In other words, we assume users choose to control higher-scoring robots with exponentially higher probability. Crucially, we also assume that the score of each robot is independent of the other robots. This makes it possible to scale the model to a large number of robots n, which would not be practical if, e.g., scores depended on interactions between the states of different robots.

User rationality model. We assume the user's control policy $\pi_H : \mathcal{S} \times \mathcal{A} \to [0, 1]$ maximizes the user's utility function R. At time t, the user chooses a robot i_H^t to control, then controls it using their policy π_H. We assume the user's scoring function ϕ maximizes the cumulative task performance of all robots:

$$\phi = \arg\max_{\phi \in \Phi} \mathbb{E} \left[\sum_{i=1}^{n} \sum_{t=0}^{T-1} R(s_i^t, a_i^t) \mid \pi_H, \pi_R \right], \tag{3.2}$$

where T is the episode horizon. The actions a_i^t are determined by whether the user was in control and executed their policy π_H, or the robot relied on its own policy π_R. Formally,

$$\mathbb{P}[a_i^t = a] = \mathbb{P}[i_H^t = i] \pi_H(a | s_i^t) + (1 - \mathbb{P}[i_H^t = i]) \pi_R(a | s_i^t), \tag{3.3}$$

where the user's choice $\mathbb{P}[i_H^t = i]$ of whether or not to control robot i is modeled in Equation 3.1.[1]

Robot policy. We assume the robot policy π_R is identical for each of the n robots, and that it does not perfectly maximize the user's utility function R. Our method is agnostic to how π_R is constructed: e.g., it could be a decision tree of hard-coded control heuristics, or a planning algorithm equipped with a forward dynamics model of the environment. In this work, we choose π_R to be a learned policy that is trained to imitate user actions. This allows us to make minimal assumptions about the task and environment, and enables the robot policy to improve as the amount and quality of user demonstration data increases.

Knowns and unknowns. We assume we know the robot policy π_R, but we do not know the user's utility function R, the user's control policy π_H, or the user's scoring function ϕ.[2]

[1] We assume the user's scoring function ϕ is optimal with respect to the utility objective in Equation 3.2 for the sake of clarity. However, our method is still useful in settings where ϕ is suboptimal. In such settings, our method will, at best, match the performance of the user's suboptimal ϕ.

[2] We assume the user's utility function R is unknown for the sake of clarity. However, our method is also useful in settings where the user's utility function R is known, but the utility-maximizing policy π_H is difficult to compute or learn from demonstrations.

Problem statement. Our assumptions about the user's rationality may not hold in settings with a large number of robots n: because the user has limited attention, they may not be able to evaluate the scores of the states of all robots at all times using their internal scoring function ϕ. As a result, they may make systematically suboptimal choices that do not maximize the expected cumulative task performance across all robots.

Our Method

Our aim is to help the user maximize the expected cumulative task performance across all robots, in settings with a large number of robots n. We do so by learning a scoring model $\hat{\phi}$ and using it to automate the user's choice of which robot to control. In conjunction, we train the robot policy π_R to imitate user action demonstrations – collected initially on a single robot, then augmented with additional demonstrations collected as the user operates each robot chosen by the scoring model $\hat{\phi}$.

We split our method into four phases. In phase one, we train the robot policy π_R using imitation learning. The user controls a single robot in isolation, and we collect a demonstration dataset $\mathcal{D}_{\text{demo}}$ of state-action pairs (s, a). Our method is agnostic to the choice of imitation learning algorithm. In phase two, we learn a scoring model $\hat{\phi}$ by asking the user to manage a small number of robots, observing which robot i_H^t the user chooses to control at each timestep based on their internal scoring function ϕ, then fitting a parametric model of ϕ that explains the observed choices. In phase three, we enable the user to manage a large number of robots by using the learned scoring model $\hat{\phi}$ to automatically choose which robot to control for them. In phase four, we update the robot's imitation policy π_R with the newly-acquired user action demonstrations from phase three.

Phase one (optional): training the robot policy π_R. In this work, we train the robot policy π_R to imitate the user policy π_H. We record state-action demonstrations $\mathcal{D}_{\text{demo}}$ generated by the user as they control one robot in isolation, and use those demonstrations to train a policy that each robot can execute autonomously during phases two and three. We implement π_R using a simple nearest-neighbor classifier that selects the action taken in the closest state for which we have a demonstration from π_H. Formally,

$$\pi_R(a|s) = \begin{cases} 1 & \text{if } (\cdot, a) = \arg\min_{(s',a') \in \mathcal{D}_{\text{demo}}} \|s - s'\|_2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

We choose a simple imitation policy for π_R in order to model real-world tasks for which even state-of-the-art robot control policies are suboptimal. Improving the autonomous robot policy π_R is orthogonal to the objective of this paper, which is to enable an arbitrary robot policy π_R to be improved by the presence of a human operator capable of intervening in challenging states.

Phase two: learning the scoring model $\hat{\phi}$. Our approach to assisting the user involves estimating their scoring function ϕ. To do so, we have the user manage a small number of robots n. While the user operates one robot using control policy π_H, the other robots take actions using the robot policy π_R trained in phase one. The user can monitor the states of all robots simultaneously, and freely choose which robot to control using their internal scoring function ϕ. We observe which robot i_H^t the user chooses to control at each timestep, and use these observations to infer the user's scoring function ϕ. In particular, we compute a maximum-likelihood estimate by fitting a parametric model $\hat{\phi} = \phi_{\boldsymbol{\theta}}$ that minimizes the negative log-likelihood loss function,

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \sum_{(s_1^t, \ldots, s_n^t, i_H^t) \in \mathcal{D}} -\phi_{\boldsymbol{\theta}}\left(s_{i_H^t}\right) + \log\left(\sum_{j=1}^{n} e^{\phi_{\boldsymbol{\theta}}\left(s_j^t\right)}\right), \tag{3.5}$$

where $\mathcal{D}$ is the training set of observed choices, and $\boldsymbol{\theta}$ are the weights of a feedforward neural network $\phi_{\boldsymbol{\theta}}$.[3] The learned scoring model $\hat{\phi}$ is optimized to explain the choices the user made in the training data, under the assumptions of the choice model in Equation 3.1. Fitting a maximum-likelihood estimate $\hat{\phi}$ is the natural approach to learning a scoring model in our setting, since the MLE can be used to mimic the user's internal scoring function and thereby assist choice in phase three, and because it can be accomplished using standard supervised learning techniques for training neural networks.

Phase three: assisted choice. At test time, we assist the user at time t by automatically switching them to controlling the robot with the highest predicted likelihood of being chosen: robot $\arg\max_{i \in [n]} \hat{\phi}(s_i^t)$. This enables the user to manage a large number of robots n, where the user is unable to evaluate the scores of the states of all robots simultaneously using their internal scoring function ϕ, but where we can trivially apply the learned scoring model $\hat{\phi}$ to the states of all robots simultaneously.

Phase four (optional): improving the robot policy π_R. While performing the task in phase three, the user generates action demonstrations (s, a) as they control each chosen robot – demonstrations that can be added to the training data $\mathcal{D}_{\text{demo}}$ collected in phase one, and used to further improve the robot policy π_R through online imitation learning. One of the aims of assisted choice in phase three is to improve the quality of these additional demonstrations, since operator interventions in challenging states may provide more informative demonstrations.

3.5 Simulation Experiments

In our first experiment, we simulate human input, in order to understand how our method performs under ideal assumptions. We seek to answer two questions: (1) does a model trained on data from a small fleet generalize to a large fleet, and (2) is our idea of treating choices as observations of preferences important for this generalization? Simulating user input enables us to assess not just the task performance of our learned scoring model, but also its ability to recover the true internal scoring function; e.g., by posing counterfactual

[3]In our experiments, we used a multi-layer perceptron with two layers containing 32 hidden units each and ReLU activations.

questions about how often the predictions made by our learned scoring model agree with the choices that would have been made by a simulated ground-truth scoring function.

Experiment Design

Setup. We evaluate our method on a custom navigation task in the DOOM environment [32]. In the navigation task, the robot navigates through a video game environment containing three linked rooms filled with hazards and health packs to reach a goal state (see screenshot in Figure 4.1 and schematic in Figure 3.4). The robot receives low-dimensional observations $s \in \mathbb{R}^4$ encoding the robot's 2D position, angle, and health, and takes discrete actions that include moving forward or backward and turning left or right. The default reward function outputs high reward for making progress toward the goal state and collecting health packs while avoiding hazards. To introduce stochasticity into the environment, we randomize the initial state of each robot at the beginning of each episode.
Simulating user input. Although simulating user input is not a part of our method, it is useful for experimentally evaluating our method under ideal assumptions. We simulate the human operator with a synthetic user policy π_H trained to maximize the environment's default reward function via deep reinforcement learning – in particular, the soft actor-critic algorithm [25]. Note that our algorithm is not aware of the utility function R, and simply treats the reinforcement learning agent π_H the same way it would treat a human user. We choose the simulated ground-truth scoring function ϕ to be the gain in value from running the user policy π_H instead of the robot policy π_R,

$$\phi(s_i^t) = V^{\pi_H}(s_i^t) - V^{\pi_R}(s_i^t), \tag{3.6}$$

where V denotes the value function, which we fit using temporal difference learning [62] on the environment's default rewards. Note that our choice of ϕ does not necessarily maximize the cumulative task performance of all robots, as assumed in Equation 3.2 in Sec. 3.4. It is a heuristic that serves as a replacement for human behavior, for the purposes of testing whether we can learn a model of ϕ that performs as well as some ground-truth ϕ. We would like to emphasize that our method does not assume knowledge of π_H or ϕ, and that the design decisions made above are solely for the purpose of simulating user input in synthetic experiments.
Manipulated variables. We manipulate the ***scoring function*** used to select the robot for the synthetic user to control. The scoring function is either (1) the ground-truth scoring function ϕ, (2) a model of the scoring function trained using our method $\hat{\phi}_{\text{luce}}$, or (3) a model of the scoring function trained using a baseline classification method $\hat{\phi}_{\text{base}}$.

Our method follows the procedure in Sec. 3.4: in phase two, it fits a scoring model $\hat{\phi}_{\text{luce}}$ to explain observations of the user's choices in a setting with a small number of robots $n = 4$, under the modeling assumptions in Sec. 3.4.

The baseline method fits a scoring model $\hat{\phi}_{\text{base}}$ that assumes a much simpler user choice model than our method: that the user selects robot i with probability $\sigma(\phi(s_i^t))$, where σ is the sigmoid function. In other words, the baseline method trains a binary classifier to distinguish between states where the user intervened and states where the user did not intervene. Unlike our method, this approach does not model the fact that the user chose where to intervene based on relative differences in scores, rather than the absolute score of each robot. Because of this modeling assumption, the baseline may incorrectly infer that a state s_i^t was not worth intervening in because the user did not select robot i at time t, when, in fact, the user would have liked to intervene in robot i at time t if possible, but ended up selecting another robot j that required the user's attention even more than robot i.

We test each scoring function in a phase-three setting with a large number of robots $n = 12$. At each timestep, we use the scoring function to choose which robot to control: the chosen robot executes an action sampled from the user policy π_H, while all other robots execute actions sampled from the robot policy π_R.

We also test each scoring function in a phase-four setting, where we re-train the robot's imitation policy π_R using the newly-acquired user demonstrations from phase three, then evaluate π_R by running it on a single autonomous robot.

Dependent measures. To measure the ***performance of the human-robot team*** in the phase-three setting with $n = 12$ robots, we compute the cumulative reward across all robots. To measure the ***predictive accuracy*** of each learned scoring model, we compute the top-1 accuracy of the robot ranking generated by the learned scoring models $\hat{\phi}_{\text{luce}}$ and $\hat{\phi}_{\text{base}}$ relative to the ranking produced by the ground-truth scoring function ϕ. To measure the ***data impact*** of each scoring model on the quality of demonstrations used to improve the robot policy π_R in phase four, we evaluate the task performance of the robot's imitation policy π_R after being re-trained on the user action demonstrations generated in phase three.

Hypothesis H1 (generalization). Our learned scoring model $\hat{\phi}_{\text{luce}}$ performs nearly as well as the ground-truth scoring function ϕ in the phase-three setting with a large number of robots, in terms of both the cumulative reward of the human-robot team and the data impact on the performance of a single robot.

Hypothesis H2 (modeling relative vs. absolute preferences). Our learned scoring model $\hat{\phi}_{\text{luce}}$ outperforms the baseline scoring model $\hat{\phi}_{\text{base}}$, in terms of all dependent measures – cumulative reward, predictive accuracy, and data impact.

Analysis.

Figure 3.2 plot the performance and data impact of each scoring function for the navigation task. In line with our hypotheses, $\hat{\phi}_{\text{luce}}$ outperforms $\hat{\phi}_{\text{base}}$ in all measures, while performing slightly worse than ϕ. In terms of predictive accuracy, we find that $\hat{\phi}_{\text{luce}}$ generalizes reasonably well: it agrees 79% of the time with the ground truth, compared to 32% for $\hat{\phi}_{\text{base}}$, which translates to better team performance. Furthermore, demonstration data from assisted choice with $\hat{\phi}_{\text{luce}}$ induces a stronger imitation policy π_R. One explanation for this result is that collecting expert action demonstrations in challenging states leads to a better imitation policy π_R than demonstrations in less challenging states. Users tend to prefer to take over robots in states that are challenging for the robot policy π_R. Our learned scoring

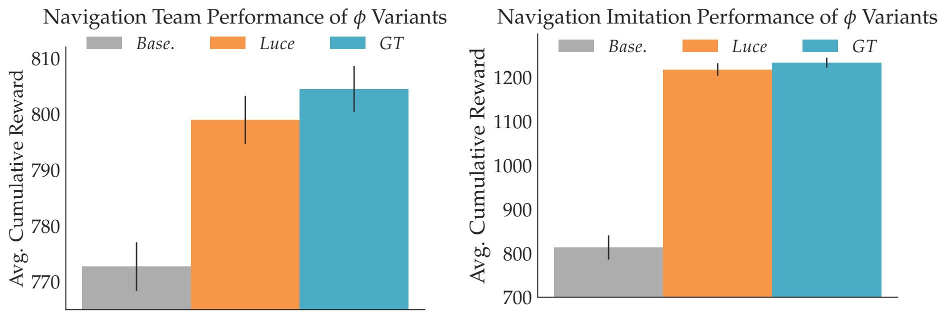

Figure 3.2: **Left**: our learned scoring model $\hat{\phi}_{\text{luce}}$ outperforms the baseline scoring model $\hat{\phi}_{\text{base}}$, while performing slightly worse than the ground-truth (GT) scoring function ϕ. Rewards are averaged across all twelve robots, and across 250 trials. **Right**: the scoring function affects which states the user demonstrates actions in during phase three, which in turn affect the performance of the imitation policy after re-training with the new demonstrations in phase four. The ground-truth (GT) scoring function ϕ leads to the best imitation performance, followed by our method $\hat{\phi}_{\text{luce}}$. The baseline $\hat{\phi}_{\text{base}}$ induces less informative demonstrations. Rewards are for a single robot policy π_R that runs without human intervention, and are averaged across 250 trials of 8 episodes each.

	Unassisted	**Assisted**	F(1,11)	p-value
Q1	2.92	4.50	17.49	$<$ **0.01**
Q2	2.25	3.92	13.75	$<$ **0.01**
Q1*	2.67	4.17	17.47	$<$ **0.01**
Q2*	2.92	3.25	0.88	0.37

Figure 3.3: Q1: it was easy to guide the robots to their goals. Q2: I was successful at guiding the robots. Q*: Q after objective measures revealed. Survey responses on a 7-point Likert scale, where 1 = Strongly Disagree, 4 = Neither Disagree nor Agree, and 7 = Strongly Agree.

model may capture that preference and, through assisted choice, allocate the user's expert actions to more challenging states. These results suggest that our assisted choice method might be useful for active learning [57] in the context of training robots via imitation [5] – one possible direction for future work.

3.6 User Study

The previous section analyzed our method's performance with synthetic data. Here, we investigate to what extent those results generalize to real user data.

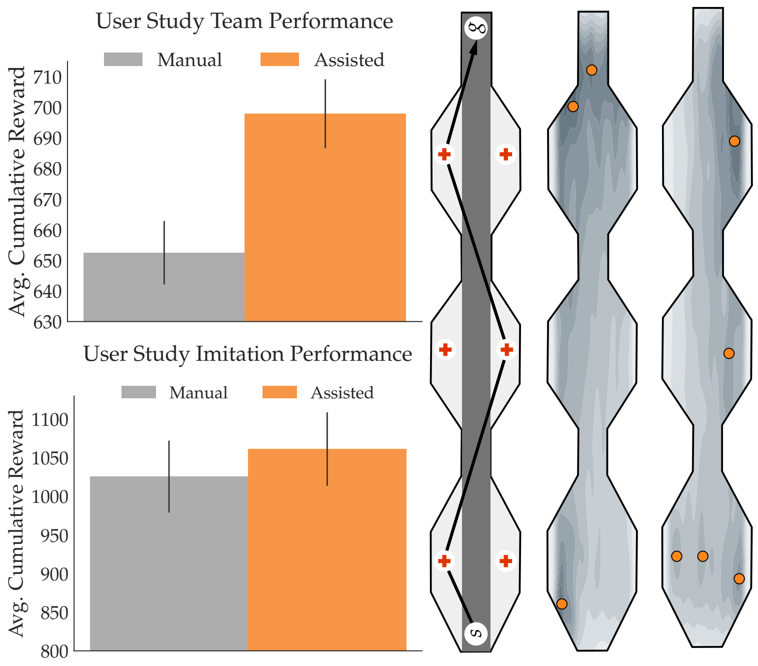

Figure 3.4: Performance of the human-robot team in phase three, averaged across twelve participants and eight trials per participant (top), and the robot's imitation policy in phase four (bottom). In the navigation task, the agent starts at s and navigates to g, while avoiding the darker gray hazard region and collecting health packs indicated by crosses. The heat maps render the learned scoring models $\hat{\phi}$ of three different users, showing the positions where the model predicts each user would most prefer to take control of the robot. Darker indicates higher predicted score, and orange circles are peaks.

Experiment Design

Setup. We use the navigation task for the study, which is split into the four phases described in Sec. 3.4. In phase one, we collect data for training our robot policy π_R via imitation: participants control a single robot – initialized with a uniform random policy – using the arrow keys on a keyboard, correcting it whenever it performs an action they would prefer it did not. We do this for ten episodes. In phase two, we collect data to train our scoring model $\hat{\phi}$: participants manage $n = 4$ robots, which they monitor simultaneously. Participants manually choose which robot to control. In phase three, participants manage $n = 12$ robots, and either manually choose which robot to control, or let the learned scoring model choose which robot to control for them. In phase four, we re-train the imitation policy π_R on the action demonstrations from phase three, and evaluate the task performance of π_R in isolation.
Manipulated variables. We manipulate whether or not the user is assisted by the learned scoring model $\hat{\phi}$ in choosing which of the $n = 12$ robots to control at any given time in

phase three. In the manual condition, the user is able to view the states of all robots simultaneously, and freely selects which robot to control using their internal scoring function ϕ. In the assisted condition, every fifteen timesteps the user is automatically switched to controlling the robot with the highest predicted likelihood of being chosen according to the learned scoring model $\hat{\phi}$.[4]

Dependent measures. As in the synthetic experiments, we measure ***performance of the human-robot team*** using the cumulative reward across all robots, and ***data impact*** using the reward achieved by a single robot running the imitation policy π_R after training on the user demonstrations generated in the phase-three setting with $n = 12$ tobots. We also conduct a survey using Likert-scale questions to measure subjective factors in the user experience, like the user's self-reported ease of use and perception of success with vs. without assisted choice. We administered this survey after each condition, once before and once after revealing the cumulative reward to the participants.

Hypothesis H3. We hypothesize that assisted choice will improve our objective and subjective dependent measures.

Subject allocation. We conducted the user study with twelve participants, nine male and three female, with a mean age of twenty-one. We used a within-subjects design and counterbalanced the order of the two conditions: manual choice, and assisted choice.

Analysis.

Objective measures. We ran a repeated-measures ANOVA with the use of assisted choice (vs. manual choice) as a factor and trial number as a covariate on the performance of the human-robot team. Assisted choice significantly outperformed manual choice ($F(1, 184) = 12.96$, $p < .001$), supporting **H3** (see left plot in Figure 3.4). Assisted choice also slightly improved the performance of the robot's imitation policy π_R in phase four (see the right plot in Figure 3.4), although the difference was not statistically significant (F(1,23)=.31, p=.59). One explanation for this result is that switching the user to a different robot every 15 timesteps was not frequent enough to produce the number of informative demonstrations needed to significantly improve π_R.

Figure 3.4 illustrates the learned scoring models $\hat{\phi}$ of two different users in the study. Each model is qualitatively different, showing that our method is capable of learning personalized choice strategies. The model for the first user predicts they prefer to take control of robots near the goal position – a strategy that makes sense for this particular user, since the autonomous robot policy trained in phase one was not capable of completing the task near the end. The models for the second user predicts they would prefer to intervene near health packs, which makes sense since the robot policies trained to imitate that user tended to fail at maneuvering close to health packs.

[4]We also tried a more flexible interface where, instead of automatically switching the user to the predicted robot, the interface continued showing all robots' states and merely highlighted the robot with the highest predicted likelihood of being chosen. The users in this pilot study tended to get confused by the suggestion interface, and preferred to be automatically switched to the predicted robot.

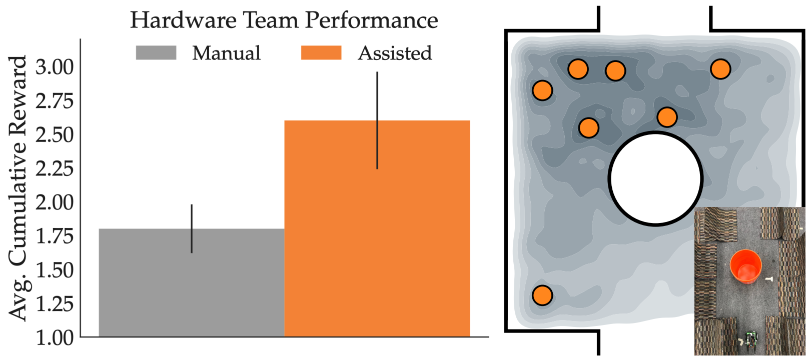

Figure 3.5: **Left**: performance of the human-robot team in phase three, averaged across five one-minute trials. **Right**: as in Figure 3.4, the heat map shows the learned scoring model.

Subjective measures. Table 3.3 shows that users self-reported that they found it easier to guide the robots to their goals and were more successful at guiding the robots in the assisted choice condition compared to the manual choice condition. The table also shows the results of running a repeated-measures ANOVA on each response. The improvement with our method was significant in all but one case: how successful users felt after we revealed their score to them.

3.7 Hardware Demonstration

The previous sections analyzed our method's performance on a simulated navigation task. Here, we investigate to what extent those results generalize to real hardware.

We evaluate our method with a single human user driving laps with up to three NVIDIA JetBot mobile robots while avoiding centrally-placed buckets and surrounding obstacles (see Figures 4.1 and 3.5). We use an OptiTrack motion capture system to measure the positions and orientations of the robots. We enable the user to control each robot's velocity – via discrete keyboard actions for driving left, right, forward, and backward – while looking through the robot's front-facing camera. The reward function outputs +1 upon each lap completion. Each episode lasts one minute.

We repeat the three phases from Sec. 3.6. In phase one, we collect 2000 state-action demonstrations for training our robot policy π_R via imitation. In phase two, we collect 1300 examples of a human's preferences (s_1^t, s_2^t, i_H^t) to train our scoring model $\hat{\phi}$: the human – an author, for the purposes of our demonstration – manages $n = 2$ robots, which they monitor simultaneously. In phase three, they manage $n = 3$ robots, and either manually choose which robot to control, or let the learned scoring model choose which robot to control for them every five seconds.

As in Sec. 3.6, we manipulate whether or not the user is assisted by the learned scoring model $\hat{\phi}$ in choosing which of the $n = 3$ robots to control at any given time in phase three. We measure ***performance of the human-robot team*** using the cumulative reward across all robots.

We find that again, assisted choice improves the performance of the human-robot team in phase three (Figure 3.5). The learned scoring model $\hat{\phi}$ helpfully predicts high scores for states close to the bucket, the boundary of the environment, or a corner where the robot tends to get stuck, where user interventions tend to be concentrated during phase two.

3.8 Limitations and Future Work

We have evaluated our method in a simulated environment, and demonstrated its feasibility on real hardware in a controlled lab setting. For complex, real-world tasks, it may be the case that learning a scoring model is as difficult as learning the user's policy. In such cases, exploiting our method's ability to gather useful demonstration data by focusing the user's attention on informative states would be an interesting area for further investigation.

Another direction for future work could be to investigate tasks where users tend to change their control policies and internal scoring models as the number of robots increases, and whether iteratively applying our method addresses non-stationary user models.

Chapter 4

Neural Decoding from BCI Feedback Alone

We think of the key,
each in his prison

The Waste Land
T.S. Eliot

4.1 Acknowledgements

The work in this section was conducted and written up by the author, Siddharth Reddy, Sergey Levine, and Anca Dragan.

4.2 Code

We believe the code for our experiments can serve as a useful starting point for further investigations so we open-source it at `https://github.com/gkswamy98/meta-rl-bci/`.

4.3 Introduction

In shared autonomy [30], user input is melded with robot action before being executed on some system. One particularly exciting class of shared autonomy systems is those based on brain-computer interfaces (BCIs) [42] because of their potential to help those with motor paralysis regain some level of autonomy. For these systems to be truly general purpose, a user should be able to freely specify what they wish the system to do rather than selecting from a predefined set of limited goals. Prior work has looked at removing the need for a pre-defined goal space via reinforcement learning [48] but required multiple sources of user input, making

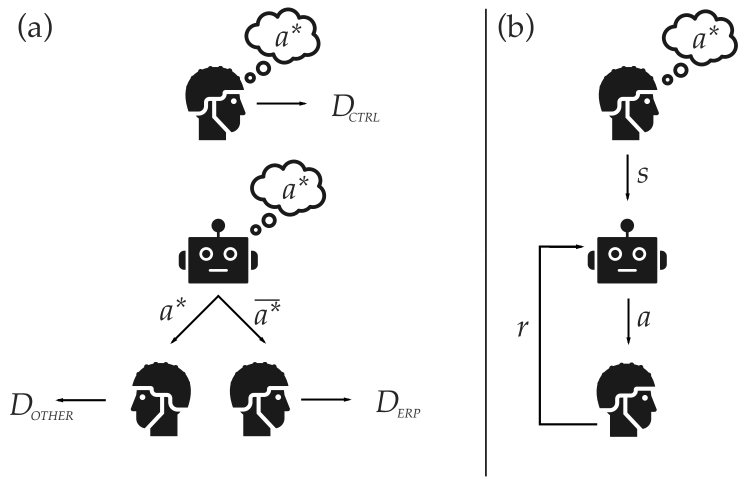

Figure 4.1: We propose a method for performing neural decoding purely from BCI feedback. We first train a baseline policy and error director (a) by having the user focus on an easily computable optimal action and then have the robot execute it or a clearly suboptimal action. We use the user's response to the robot's action to train an error classifier. Then, we fine-tune our policy using the trained error classifier as a reward signal for reinforcement learning (b).

it difficult for those with limited motor control to benefit from such an approach. Instead, we propose a system that is able to learn to decode user inputs purely from neural feedback. Our approach has two phases. In the first phase, we automatically generate tasks in which the user can easily imagine the optimal action at each timestep. The system computes the optimal action and a clearly suboptimal action, then randomly samples one of them and executes it. We record the user's neural response to the action, and train a binary error classifier to predict whether or not the most recent action was optimal given the observed features in the user's neural response signal. Then, we treat the labels predicted by the error classifier as rewards, and fine-tune our control policy to the user's desired task via off-policy maximum entropy deep reinforcement learning.

4.4 Related Work

A general challenge with data gathered from BCIs is the change over time of a single user's background brain noise and the difference across users of this noise. This factor also makes it difficult to collect data iteratively over time to improve a policy [50] mapping from brain states to actions because of the shift in state distribution over time. Instead, we treat past datasets as tasks from a distribution over possible neural decoding problems and apply a meta-learning approach to learn network initializations that can quickly adapt to new datastreams over time [43]. We believe this facet of our approach is crucial to reducing the warm-up time required every time the headset is removed as well.

Our work on robot error detection builds on the detection of event-related potentials (ERPs) from BCIs [60]. To have clean data for training a classifier, it is important that during data collection, errors occur infrequently enough that the subject does not feel fatigue or start to mix events together. This is known as the "oddball paradigm" and we borrow the standard parameters for it [29].

For detecting controls, we turn to steady-state visually evoked potentials (SSVEPs) for their excellent signal-to-noise ratio [28]. When a user focuses on an area of a screen that pulses at a certain frequency (usually in the 7 Hz to 14 Hz range), a standing wave is produced that is clearly detectable by electrodes located near the back of a user's head. These waves are distinct and strong enough to enable users to spell out words purely by focusing on different areas of the screen [14]. We use a separate frequency for each control action a person could want to take.

To detect intended user controls and ERPs, we use the EEGNet architecture, both for its performance and ease of being incorporated into gradient-based meta-learing and deep reinforcement learning approaches [36]. For SSVEP detection, we use the modified parameters from [64]. Because of the relatively limited amount of data we collect and to prevent overfitting, we scale down parameters to a fraction of their prescribed values while keeping the relative ratios constant.

In the style of Reddy et al. [48], we use deep reinforcement learning to fine-tune our control policy. Because the reward feedback decoded from BCI signals will be noisy, we adopt a maximum entropy approach [67]. To not waste user time doing on-policy updates during training, we adopt an off-policy maximum entropy deep reinforcement learning algorithm, Soft Actor Critic [25]. Prior work has looked at using ERP based feedback to assist reinforcement learning agents [3], [66]. However, these approaches have the user watch an agent that does not attempt to map from their brain states to actions – controls are instead autonomously generated or delivered via some mechanism like a joystick.

Unfortunately because of issues with our hardware, the experiments we present in this section are done via simulated BCI input. We borrow the methods and parameters of Krol et al. for simulating ERPs [35] and add in phase-shifted sine waves at the appropriate frequency for simulating SSVEPs.

4.5 Bootstrapping Human Feedback for Training Neural Decoders

Interface Design

To generate data for training our error detector and control policy, we designed an interface that was then implemented in PyGame [58]. We focus on 2D cursor control because of the applicability to a wide variety of computer-based tasks but believe a similar method could

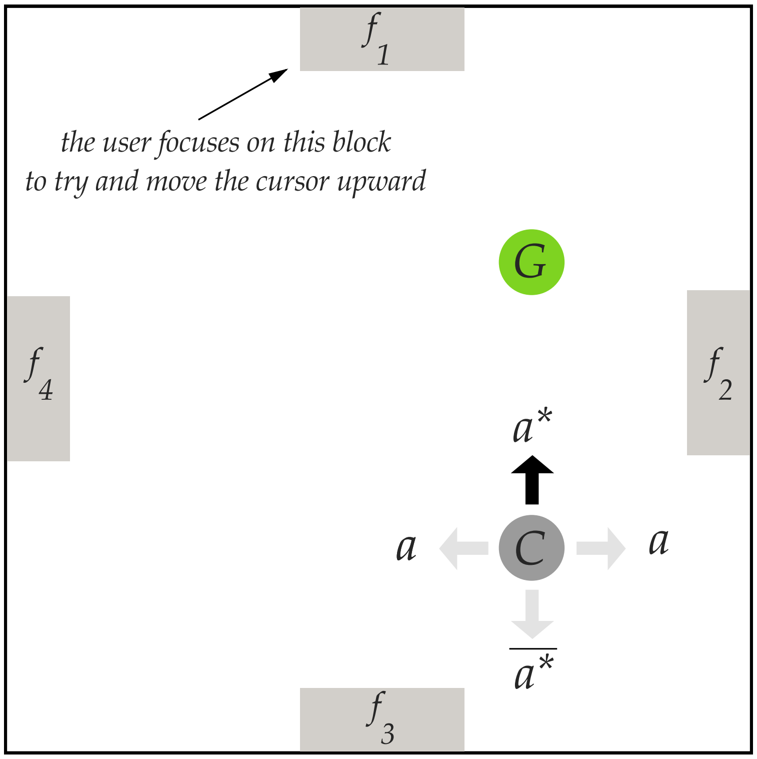
f_1
the user focuses on this block
to try and move the cursor upward
G
f_4
f_2
a^*
a
C
a
$\overline{a^*}$
f_3

Figure 4.2: A schematic of our user interface. Goals are positions on the screen and the user can signal to move the cursor up, down, left, or right by focusing on the corresponding block, each of which pulse at a distinct frequency.

be used for more specific tasks like typing. To move the cursor in a particular direction, the user focuses visually on the corresponding control block (light grey in Fig. 4.2). Each block pulses at a different frequency and has a different color (omitted for clarity in the schematic) – we choose $f_1 = 8.25$, $f_2 = 10.25$, $f_3 = 12.25$, $f_4 = 14$ (all measured in Hertz) to space out frequencies in the detectable range and prevent low least common multiples that can show up as harmonics in the detected Fourier spectrum. We generate goals by randomly sampling points within the bounds of the screen. To allow the user to both respond to execution errors and think of the next action to take, it is critical to space out robot actions temporally [66], [3]. The cursor moves every 2.5 seconds with the data from the earlier part of the interval being used to train the error classifier and that from the latter being used to train the control policy.

Method

Our method (Algorithm 1), consists of three phases that build upon each other to produce a control policy that is well fit to both the user's current mental state and their desired goal distribution.

Phase 1: Meta-learning from past data. Because of the shifts in background brain noise, user attention, headset position, a trained neural decoder that works very well one day may not be able to perform nearly as well the next day. However, given the structure present in the data, using it to find a network initialization that enables sample-efficient learning might allow users to have to spend less and less time configuring decoding the neural interface. To accomplish this task, we use REPTILE [43], a first-order gradient-based meta-learning method that is easy to implement. We train the initialization for the reward decoder R via minimizing the cross-entropy loss on the previously collected datasets of error-related and non-error related potentials, $D_{1..n-1}^{ERP}$. We perform the same procedure for the policy π on past control-related datasets, $D_{1..n-1}^{CTRL}$. For initializing the state-action value function Q, we use $D_{1..n-1}^{CTRL}$ and a mean-squared error loss where the target is the one-hot vector of the correct action – our episodes are only a single action long and we give +1 reward for the correct action and 0 otherwise so this is a sufficient initialization.

Phase 2: Training via imitating an optimal controller. Because ERPs are only accurate if they occur infrequently enough, we need to train the policy sufficiently well before we can fine-tune it to a user's desired task via reinforcement learning. We do this by choosing simple tasks where optimal actions are easy for the user to compute, and ask the user to imagine the optimal action a^*. We add their recorded signals here to D_n^{CTRL} with label a^*. The robot then executes the optimal action a^* with probability $1 - p_{error}$, or with probability p_{error} executes a clearly suboptimal action (e.g., going away from the goal). We set $p_{error} = 0.2$. We then record the user's neural response to this action via BCI and create a dataset of responses to optimal and suboptimal actions, D_n^{ERP}. These datasets are then used to train π, R, Q using the same loss functions as above.

Phase 3: Fine-tuning via reinforcement learning. Because the labels generated by R are noisy, we adopt a maximum entropy reinforcement learning (MaxEnt RL) approach [67]. We would like to be able to perform off-policy training without the user in the loop, so as to reduce the number of user interactions and not force the user to constantly adapt to a changing policy. We use the TF2RL implementation of the Soft Actor Critic algorithm to perform off-policy MaxEnt RL [45]. At each timestep, the policy takes in user brain state s and decodes an action a. The action is then executed. The user responds, and we decode a reward by applying R to the data streamed from the BCI. We then get the next state from the user for control decoding, s'. These (s, a, r, s') tuples are added to a replay buffer B that is used as the data source for Soft Actor-Critic. In contrast to the datasets we collect in Phase 2, we do not save these datapoints because of the noise in r that might harm the initialization of future decoders.

Algorithm 1: Neural Decoding from BCI Feedback Alone

```
Data: $D^{CTRL}_{1..n-1}$, $D^{ERP}_{1..n-1}$: past datasets, BCI(·): live data filtered and denoised
Result: $\pi$: policy mapping from brain states to control actions
/* Phase 1:  Meta-learning from past data                                */
$\pi \leftarrow \text{REPTILE}(D^{CTRL}_{1..n-1}, \text{loss} = \text{cross-entropy})$ ;
$Q \leftarrow \text{REPTILE}(D^{CTRL}_{1..n-1}, \text{loss} = \text{MSE})$ ;          // Use one-hot labels
$R \leftarrow \text{REPTILE}(D^{ERP}_{1..n-1}, \text{loss} = \text{cross-entropy})$ ;
/* Phase 2:  Training via imitating an optimal controller                */
$D^{CTRL}_n, D^{ERP}_n = \{\}, \{\}$;
while under time limit do
    $D^{CTRL}_n = D^{CTRL}_n \cup \text{BCI}(\cdot)$ ;
    Compute optimal action $a*$;
    if $x \sim U[0, 1] \leq p_{error}$ then
        $a = \bar{a^*}$ ;                      // Take action that is clearly suboptimal
    else
        $a = a^*$
    end
    Execute $a$ ;
    $D^{ERP}_n = D^{ERP}_n \cup \text{BCI}(\cdot)$ ;
end
Train $\pi$ on $D^{CTRL}_n$ via supervised learning ;
Train $Q$ on $D^{CTRL}_n$ via supervised learning ;
Train $R$ on $D^{ERP}_n$ via supervised learning ;
Save $D^{CTRL}_n$ and $D^{ERP}_n$ ;
/* Phase 3:  Fine-tuning via reinforcement learning                      */
$B = \{\}$ ;
while under time limit do
    $s \leftarrow BCI(\cdot)$;
    $a \leftarrow \pi(s)$;
    Execute $a$ ;
    $r \leftarrow R(BCI(\cdot))$;
    $B \leftarrow B \cup (s, a, r)$ ;
end
Train $\pi, Q$ on $B$ via Soft Actor-Critic ;
```

4.6 Simulated Experiments

Unfortunately, due to issues with our hardware, we were only able to conduct simulated experiments to test our previously described method.

Simulator Design

To enforce some level of realism in our experiments, we designed our simulator to have parameters close to that of the expected real-world data. We use Brainflow [46], an open-source framework that enables both easy streaming from hardware boards and simulating brain background noise as the bedrock for our simulations. The data is streamed at 250 Hz across 8 channels. After grabbing noise from the simulated board, we add in the signal (both SSVEPs and ERPs). For SSVEPs, we generate a sine wave at the appropriate frequency and add it into a single channel of the sample (because the SSVEP is a local effect near the back of the head). We randomly shift the phase for each each generated sample. We choose a signal-to-noise-ratio (SNR) much lower than that seen in real-life systems. For ERPs, we build upon the work of Krol et al. [35] and simulate the ERP as the sum of several scaled and shifted Gaussian PDFs. We simulate the P1, N1, and P3 peaks of a standard ERP. We add this waveform (scaled to the same SNR as Krol et al.) to $\frac{3}{4}$ of the channels because ERPs are a global effect. After adding in the signal, we perform task-specific filtering and denoising before training our networks. For ERP detection, we bandpass filter data between 0.5 Hz and 40 Hz before performing rolling median and wavelet-based denoising. We also scale each sample by the inverse of the per-channel standard deviation. For control decoding, we bandpass filter data between 6 Hz (to eliminate spurious harmonics) and 50 Hz (to preserve legitimate harmonics) and then perform the same denoising. Note that the denoising is performed per-task – because of the difference in input distributions for the tasks, applying a rolling filter before segmenting each sample into the per-task intervals would severely harm task performance. For both tasks, we found this filtering and denoising to be critical for good performance and avoiding overfitting. We collect 30 minutes of data for training the baseline models and 15 minutes of data for fine tuning.

Analysis

We were able to collect limited datasets on BCI hardware (Fig. 4.3) which seem to indicate that as we suspected, there is limited transfer across sessions for detecting ERPs. Because of how similar the generated signals are between datasets, we did not have the need to use a meta-learned initialization for our simulated experiments. Akinola et al. note that they need around a 60 percent ERP decoding accuracy for neural feedback to be useful in training Deep RL agents [3]. However, because of the complexity of the task we are performing, we noted we needed above 40 percent control policy task accuracy before fine-tuning and above 75 percent ERP detection accuracy to be able to successfully train an agent. This is well within the accuracies reported for these tasks by other sources ([66], [64]). The reinforcement learning results are only from trials that satisfied these criterion. For fine-tuning with reinforcement learning, to test the adaptive capabilities of our method, we focus on a subset of possible goals and cursor start positions that require the cursor to move upwards and to the left only, rather than the full set of start and end positions. Because of this, we do not freeze any weights but if the user's goal distribution is sufficiently close to the task distribution in the

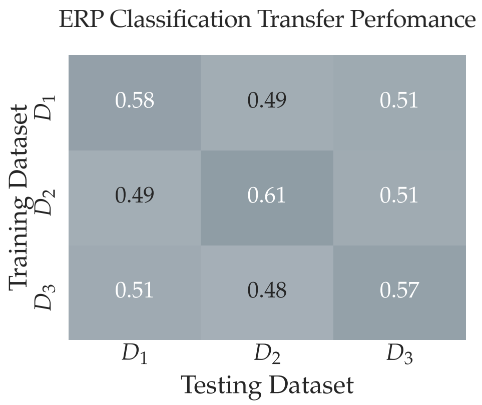

Figure 4.3: Test accuracy for a network trained on one balanced dataset and tested on another. While the main diagonal is around 60, the off-diagonal elements are around a half, indicating limited transfer given this is a binary classification problem.

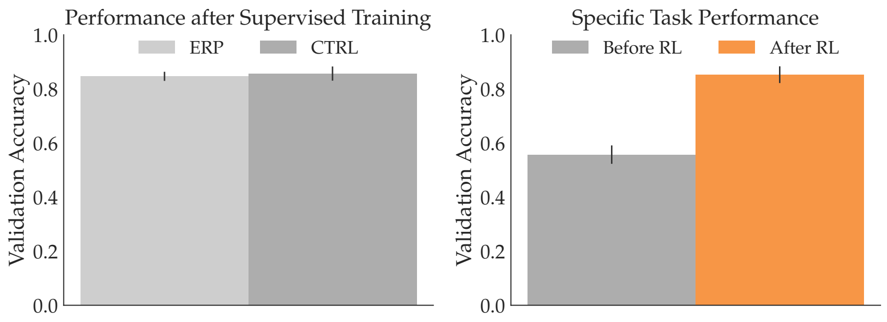

Figure 4.4: On the left, we display validation performance after supervised training of R (ERP classifier) and π (control decoder). On the right, we display task performance before and after reinforcement learning from simulated human feedback. All of these results are generated on at least four independent datasets.

previous stage, to make sure the noisy labels do not harm the existing featurizers, we would recommend training only the fully-connected parts of the EEGNets.

As we can see in Fig. 4.4, we are able to effectively decode ERP feedback on our collected datasets as well as controls. We match the performance on these tasks seen by other groups ([66], [64]). We also see a significant improvement in user-specific task performance after fine tuning via RL.

4.7 Limitations and Future Work

Our preceding results are based purely on simulated experiments so a controlled user study is among the most immediate of our next steps in more problem domains. We also focus on discrete action spaces and it is not immediately apparent how one could generalize this method to continuous spaces due to needing to visually render each of the SSVEP-inducing boxes. However, even for discrete action spaces, the curse of dimensionality might make our method difficult to apply for sufficiently high-dimensional tasks. We are therefore interested in combining our work with that of Jeon et al. to scale to more complex shared autonomy scenarios [31].

Chapter 5

Conclusions

Zeal without prudence is like a ship adrift.

A Portrait of the Artist as a Young Man
James Joyce

5.1 Summary

In the preceding three chapters, we presented experimental evidence and methods for dealing with the subtleties of learning with humans in the loop. We explored the benefits of assuming structure in behavior even when it is not totally accurate, an algorithm for learning generalizable models of user behavior by modeling their decision process, and described a proof-of-concept system for seamless human-machine collaboration. As discussed in each chapter, these results should be taken with a grain of salt – none of these algorithms are field-deployable but we believe they contain key insights that, with sufficient replication and more extensive experiments, could serve as a useful foundation for building robots that are, by design, more considerate in their interactions with people.

5.2 On the Impact of Our AI

All of the methods described in this thesis, at some level, require data to be collected from people. Beyond the application-specific considerations we focused on previously, the general idea of collecting data from people brings up concerns of privacy and the centralization of power. While these issues are outside the scope of this thesis, we wish to briefly highlight their importance and relevance to the successful and mutually beneficial deployment of systems based on the preceding ideas.

Making sure that the data collected from users for training learning systems remains secure is important both for personal safety and widespread adoption. We refer interested readers to the work of Stoica et al. for an overview of challenges and approaches to dealing with the systems side of the above algorithms [61]. While movement data like that required for the algorithm described in Chapter 2 might be sufficiently general to be easily anonymizable, EEG data like that used in the algorithm in Chapter 4 is incredibly personal and, if linkable to a specific person, could in theory be used to remove the veil that protects our thoughts from being read [27]. There is limited work on security for brain-computer interfaces but we hope that as the technology becomes more powerful, there is more focus on making sure it is used responsibly, building on preliminary work like that of Li et al. [39].

Another ethical concern we wish to highlight is that with better models of user behavior, the ongoing efforts of large corporations to monetize attention [65] may become even more successful and lead to even higher levels of concentration of power and data (the two being linked via said models). However, we do not believe any of the methods in this thesis could be directly be applied towards such goals.

Lastly, we would like to flag a potentially hazardous application of the method described in Chapter 3: the scaling of human operator attention for supervising fleets of semi-autonomous weaponized machines. While we believe significant work would be needed to apply such a method to such an application, we would like those who build upon our work to consider the broader impacts of what they are contributing to. In the words of John Donne:

No man is an island.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. “Apprenticeship learning via inverse reinforcement learning”. In: *ICML ’04: Proceedings of the twenty-first international conference on Machine learning*. Banff, Alberta, Canada: ACM, 2004, p. 1. ISBN: 1-58113-828-5.

[2] Peter Aigner and Brenan McCarragher. “Human integration into robot control utilising potential fields”. In: *Proceedings of International Conference on Robotics and Automation*. Vol. 1. IEEE. 1997, pp. 291–296.

[3] Iretiayo Akinola et al. *Accelerated Robot Learning via Human Brain Signals*. 2019. arXiv: `1910.00682 [cs.RO]`.

[4] Galen Andrew and Jianfeng Gao. “Scalable Training of L1-Regularized Log-Linear Models”. In: *Proceedings of the 24th international conference on Machine learning (ICML)*. Corvalis, Oregon, 2007, pp. 33–40.

[5] Brenna D Argall et al. “A survey of robot learning from demonstration”. In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[6] Haoyu Bai et al. “Intention-aware online POMDP planning for autonomous driving in a crowd”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 454–460.

[7] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. “Action understanding as inverse planning”. In: *Cognition* 113.3 (2009), pp. 329–349.

[8] Gary S Becker. *The economic approach to human behavior*. University of Chicago press, 2013.

[9] Alexander Broad, Todd Murphey, and Brenna Argall. “Highly Parallelized Data-driven MPC for Minimal Intervention Shared Control”. In: *arXiv preprint arXiv:1906.02318* (2019).

[10] David J Bruemmer, Donald D Dudenhoeffer, and Julie L Marble. “Dynamic-Autonomy for Urban Search and Rescue.” In: *AAAI mobile robot competition*. 2002, pp. 33–37.

[11] Baptiste Busch et al. “Learning Legible Motion from Human–Robot Interactions”. In: *International Journal of Social Robotics* (2017), pp. 1–15.

[12] Micah Carroll et al. *On the Utility of Learning about Humans for Human-AI Coordination*. 2019. arXiv: `1910.05789 [cs.LG]`.

[13] Peter Carruthers and Peter K Smith. *Theories of theories of mind*. Cambridge University Press, 1996.

[14] Xiaogang Chen et al. "High-speed spelling with a noninvasive brain–computer interface". In: *Proceedings of the National Academy of Sciences* 112.44 (2015), E6058–E6067. ISSN: 0027-8424. DOI: `10.1073/pnas.1508080112`. eprint: `https://www.pnas.org/content/112/44/E6058.full.pdf`. URL: `https://www.pnas.org/content/112/44/E6058`.

[15] Robert Cohn, Edmund Durfee, and Satinder Singh. "Comparing action-query strategies in semi-autonomous agents". In: *Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011.

[16] M Bernardine Dias et al. "Sliding autonomy for peer-to-peer human-robot teams". In: (2008).

[17] Anca D. Dragan. *Robot Planning with Mathematical Models of Human State and Action*. 2017. arXiv: `1705.04226 [cs.RO]`.

[18] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. "Legibility and predictability of robot motion". In: *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*. IEEE Press. 2013, pp. 301–308.

[19] Anca D Dragan and Siddhartha S Srinivasa. "A policy-blending formalism for shared control". In: *The International Journal of Robotics Research* 32.7 (2013), pp. 790–805.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. arXiv: `1703.03400 [cs.LG]`.

[21] Chelsea Finn, Ian Goodfellow, and Sergey Levine. "Unsupervised learning for physical interaction through video prediction". In: *Advances in neural information processing systems*. 2016, pp. 64–72.

[22] György Gergely and Gergely Csibra. "Teleological reasoning in infancy: The naıve theory of rational action". In: *Trends in cognitive sciences* 7.7 (2003), pp. 287–292.

[23] Jan Gläscher et al. "States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning". In: *Neuron* 66.4 (2010), pp. 585–595.

[24] Alison Gopnik and Henry M Wellman. "10 The theory theory". In: *Mapping the mind: Domain specificity in cognition and culture* (1994), p. 257.

[25] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. Tech. rep. 2018.

[26] Ashley Hill et al. *Stable Baselines*. 2018.

[27] Marcello Ienca and Pim Haselager. "Hacking the brain: brain–computer interfacing technology and the ethics of neurosecurity". In: *Ethics and Information Technology* 18.2 (2016), pp. 117–129.

[28] Zafer İşcan and Vadim V Nikulin. "Steady state visual evoked potential (SSVEP) based brain-computer interface (BCI) performance under different perturbations". In: *PloS one* 13.1 (2018).

[29] Iñaki Iturrate et al. "Teaching brain-machine interfaces as an alternative paradigm to neuroprosthetics control". In: *Scientific reports* 5 (2015), p. 13893.

[30] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. "Shared autonomy via hindsight optimization". In: *arXiv preprint arXiv:1503.07619* (2015).

[31] Hong Jun Jeon, Dylan P. Losey, and Dorsa Sadigh. *Shared Autonomy with Learned Latent Actions*. 2020. arXiv: `2005.03210 [cs.RO]`.

[32] Michał Kempka et al. "ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning". In: *IEEE Conference on Computational Intelligence and Games*. The best paper award. Santorini, Greece: IEEE, Sept. 2016, pp. 341–348. URL: `http://arxiv.org/abs/1605.02097`.

[33] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[34] David Kortenkamp, Debra Keirn-Schreckenghost, and R Peter Bonasso. "Adjustable control autonomy for manned space flight". In: *2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484)*. Vol. 7. IEEE. 2000, pp. 629–640.

[35] Laurens R. Krol et al. "SEREEGA: Simulating Event-Related EEG Activity". In: *bioRxiv* (2018). DOI: `10.1101/326066`. eprint: `https://www.biorxiv.org/content/early/2018/05/18/326066.full.pdf`. URL: `https://www.biorxiv.org/content/early/2018/05/18/326066`.

[36] Vernon J Lawhern et al. "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". In: *Journal of neural engineering* 15.5 (2018), p. 056013.

[37] Sang Wan Lee, Shinsuke Shimojo, and John P O'Doherty. "Neural computations underlying arbitration between model-based and model-free learning". In: *Neuron* 81.3 (2014), pp. 687–699.

[38] Sergey Levine and Vladlen Koltun. "Continuous Inverse Optimal Control with Locally Optimal Examples". In: *CoRR* abs/1206.4617 (2012). arXiv: `1206.4617`. URL: `http://arxiv.org/abs/1206.4617`.

[39] QianQian Li, Ding Ding, and Mauro Conti. "Brain-computer interface applications: Security and privacy challenges". In: *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2015, pp. 663–666.

[40] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[41] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[42] Katharina Muelling et al. *Autonomy Infused Teleoperation with Application to BCI Manipulation.* 2015. arXiv: `1503.05451 [cs.RO]`.

[43] Alex Nichol, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms". In: *arXiv preprint arXiv:1803.02999* (2018).

[44] Stefanos Nikolaidis and Julie Shah. "Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy". In: *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction.* IEEE Press. 2013, pp. 33–40.

[45] Kei Ohta. *TF2RL.* `https://github.com/keiohta/tf2rl`. 2019.

[46] OpenBCI. *Brainflow.* `https://github.com/OpenBCI/brainflow`. 2019.

[47] C. Perez. *Predictive Learning is the New Buzzword in Deep Learning.* URL: `https://medium.com/intuitionmachine/predictive-learning-is-the-key-to-deep-learning-acceleration-93e063195fd0`.

[48] Siddharth Reddy, Sergey Levine, and Anca Dragan. "Shared Autonomy via Deep Reinforcement Learning". In: *arXiv preprint arXiv:1802.01744* (2018).

[49] Bradley Rhodes and Thad Starner. "Remembrance Agent: A continuously running automated information retrieval system". In: *The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology.* 1996, pp. 487–495.

[50] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* 2011, pp. 627–635.

[51] Dorsa Sadigh et al. "Planning for Autonomous Cars that Leverage Effects on Human Actions". In: *Proceedings of Robotics: Science and Systems.* RSS '16. Ann Arbor, Michigan, 2016.

[52] Edward Schmerling et al. "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction". In: *arXiv preprint arXiv:1710.09483* (2017).

[53] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[54] John Schulman et al. "Trust region policy optimization". In: *International Conference on Machine Learning.* 2015, pp. 1889–1897.

[55] Wilko Schwarting et al. "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention". In: *2017 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. 2017, pp. 1928–1935.

[56] Brennan Sellner, Reid Simmons, and Sanjiv Singh. "User modelling for principled sliding autonomy in human-robot teams". In: *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III.* Springer, 2005, pp. 197–208.

[57] Burr Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[58] Pete Shinners. *PyGame*. `http://pygame.org/`. 2011.

[59] Beate Sodian, Barbara Schoeppner, and Ulrike Metz. "Do infants apply the principle of rational action to human agents?" In: *Infant Behavior and Development* 27.1 (2004), pp. 31–41.

[60] Martin SpÃŒler and Christian Niethammer. "Error-related potentials during continuous feedback: using EEG to detect errors of different type and severity". In: *Frontiers in Human Neuroscience* 9 (2015), p. 155. ISSN: 1662-5161. DOI: `10.3389/fnhum.2015.00155`. URL: `https://www.frontiersin.org/article/10.3389/fnhum.2015.00155`.

[61] Ion Stoica et al. *A Berkeley View of Systems Challenges for AI*. 2017. arXiv: `1712.05855 [cs.AI]`.

[62] Gerald Tesauro. "Temporal difference learning and TD-Gammon". In: *Communications of the ACM* 38.3 (1995), pp. 58–68.

[63] Stephen Tu and Benjamin Recht. "Least-squares temporal difference learning for the linear quadratic regulator". In: *arXiv preprint arXiv:1712.08642* (2017).

[64] Nicholas Waytowich et al. "Compact convolutional neural networks for classification of asynchronous steady-state visual evoked potentials". In: *Journal of neural engineering* 15.6 (2018), p. 066031.

[65] T. Wu. *The Attention Merchants: The Epic Scramble to Get Inside Our Heads*. Vintage Books, a division of Penguin Random House LLC, 2017. ISBN: 9780804170048. URL: `https://books.google.com/books?id=mNkxDwAAQBAJ`.

[66] Duo Xu et al. "Playing Games with Implicit Human Feedback". In: 2020.

[67] Brian D. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 978-1-57735-368-3. URL: `http://dl.acm.org/citation.cfm?id=1620270.1620297`.

[68] Brian D Ziebart et al. "Planning-based prediction for pedestrians". In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE. 2009, pp. 3931–3936.