Enabling Generalization of Human Models for Human-AI Collaboration to New Tasks

Xiaocheng Yang
Anca Dragan

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-199
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-199.html

August 13, 2021

Enabling Generalization of Human Models for Human-AI Collaboration to New Tasks

by Xiaocheng (Mesut) Yang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Anca Dragan
Research Advisor

(Date)

* * * * * * *

Professor Satish Rao
Second Reader

(Date)

Enabling Generalization of Human Models for Human-AI Collaboration to New Tasks

by

Xiaocheng (Mesut) Yang

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Anca Dragan, Chair
Professor Satish Rao

Summer 2021

Enabling Generalization of Human Models for Human-AI Collaboration to New Tasks

Abstract

Enabling Generalization of Human Models for Human-AI Collaboration to New Tasks

by

Xiaocheng (Mesut) Yang

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Anca Dragan, Chair

Human modeling is a crucial step for achieving good human-AI collaboration, and human data provides us with information on human behavior and thus plays an important role in the process. Even though existing methods work well on a single task with the help of plenty of on-task human data, real-world human-AI collaborations usually involve a distribution of disjoint tasks, and collecting human data on every single task is unrealistic. Consequently, naive human modeling could fail in tasks without human data. However, as long as we know the distribution of tasks, we can still use self-play to obtain a multi-task self-play policy. Since this policy will need to learn robust representations of all tasks, it can serve as an effective initialization for human models. We provide theoretical justification for this technique, and show its benefits on a challenging multi-task setting: multi-layout *Overcooked-AI*.

Contents

List of Figures

List of Tables

Acknowledgments

I would first like to thank my research mentor Micah Carroll for helping me learn everything I need to be a good researcher: from codebase management practice to paper-reading techniques. He is so knowledgeable that I look forward to every meeting because I know I will always learn something. Aside from being an extremely intellectual and helpful collaborator, he is, most importantly, a great friend.

Next, I would like to thank my advisor Professor Anca Dragan for giving me an opportunity to work in this exciting field of human-AI interaction. It was a dream-come-true for me to work with the pioneer in a field, and I really appreciate all the guidance and advice.

In addition, I would like to thank Professor Rao and all my fellow CS 188: Intro to AI TAs (including but not limited to Albert, Cathy, Carl, and Regina) for all the fascinating AI-related conversations in between CS 188 commitments. They have some of the brightest minds and warmest hearts at Berkeley, and I wouldn't have been where I am without them.

Last but not least, I would like to thank my parents for their love and motivation. Their life experiences inspired me to pursue graduate school, as both of them came from nothing yet became among the best in their field through hard-work and dedication. They are the best role models I could possibly ask for.

Chapter 1

Introduction

Even though AI systems have been able to solve increasingly complex games, competitive games [10][14][3] have received a lot more attention than collaborative ones. One of the main reasons is that building collaborative AI agents requires modeling the collaborator, which poses a unique challenge especially when they are human.

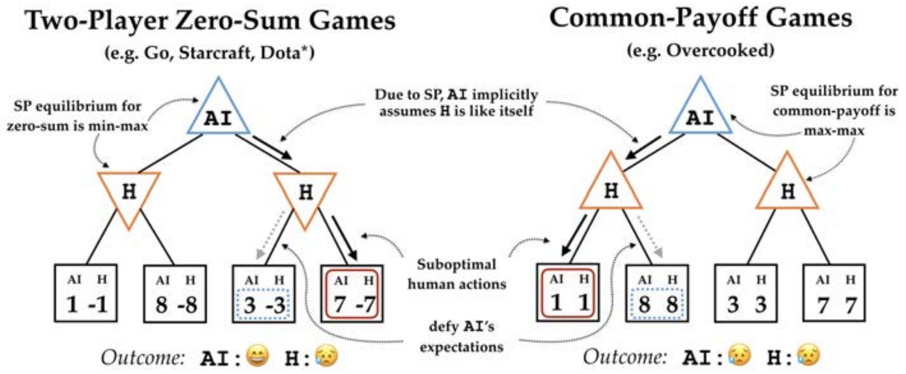

Two-Player Zero-Sum Games
(e.g. Go, Starcraft, Dota*)
Common-Payoff Games
(e.g. Overcooked)
SP equilibrium for zero-sum is min-max
Due to SP, AI implicitly assumes H is like itself
SP equilibrium for common-payoff is max-max
AI
H
AI H
1 -1
8 -8
3 -3
7 -7
1 1
8 8
3 3
7 7
Suboptimal human actions
defy AI's expectations
Outcome: AI: 😄 H: 😥
Outcome: AI: 😥 H: 😥

Figure 1.1. A figure from previous work [4] that illustrate the challenge of collaborative games

One way to explicitly model a human collaborator (proposed in previous work[4]) is to collect data on human-human collaboration and construct human models offline with imitation learning. This method is proven to work well when we know the exact task that the collaborative AI agents will be deployed in, and we can collect a large amount of human data on that specific task. However, when we only have access to the distribution of deployment tasks but cannot collect data on all of them, current human modeling techniques struggle to produce satisfying results because of the challenging generalization required.

Generalization in machine learning is usually measured by performance in unseen situations. A human model that is good at generalization should be a good proxy for the human not only on tasks for which we have human data for, but also those for which we do not have human data for. Even though there have been significant advancements in improving generalization for reinforcement learning (CURL, RAD), these have not been applied widely to the context of imitation learning. In this work, we are interested in investigating an imitation-learning-specific technique for improving generalization.

Even though low-capacity human models, including but not limited to planning[11], Boltzmann[1][13] and theory-of-mind[9][8], are popular techniques that have been extensively studied, we focus on improving the alternative option: end-to-end high-capacity human models that utilize neural networks. We investigate ways to improve training outcome when using a high-capacity human models to obtain good generalization performance.

The rest of the thesis is organized as follows: In Chapter 2, we give the mathematical definitions for important background concepts in human modeling and multi-task generalization. In Chapter 3, we formally introduce the human model generalization problem and specify the objective. In Chapter 4, we investigate the challenges of optimizing for generalization in human modeling, and provide theoretical justifications for why initializing with self-play policies addresses the challenges. In Chapter 5, we introduce multi-layout *Overcooked-AI*, the evaluation framework, and all the human modeling conditions. In Chapter 6, we show that initializing with self-play policies indeed helps us obtain a better human model. Lastly, in Chapter 7, we summarize our findings and discuss future directions. Overall, we describe our contributions as:

- Formalizing the generalization problem in human modeling, and implementing multi-layout *Overcooked-AI*, an experimental framework to evaluate methods attempting to address this problem.
- Identifying multi-task self-play policy as a promising initialization for generalize human models.
- Demonstrating that initializing with multi-task self-play policies helps improve human model generalization capabilities to new tasks, providing better collaborative AI agents.

Chapter 2

Background

2.1 Collaborative Games

We consider a symmetrical two-player collaborative game $\mathcal{G}$ given by tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_{init})$, where $\mathcal{S}$ is the set of all possible states in the game, $\mathcal{A}$ is the set of all possible actions for each player, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{A} \to [0,1]^{|\mathcal{S}|}$ is the transition probability function that, given the initial state, current actions of both players, outputs a transition probability over the destination state, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ is the reward function that outputs the joint reward for both agents given the state and the joint action, and p_{init} specifies the initial state distribution.

Let $\Pi : \mathcal{S} \to [0,1]^{|\mathcal{A}|}$ be policies, the class of functions that given a state, output a probability distribution over all legal actions. Let Ξ be a space of trajectories for a collection of state, joint action $(\{(s, a_{joint})\})$

Let $\mathcal{C} : \mathcal{G} \times \Pi \times \Pi \to \Xi$ be stochastic operator for collecting trajectories from a particular task with a specific pair of policies.

Last but not least, let $\rho : [\mathcal{G}] \times \Pi \times \Pi \to \mathbb{R}$ be an operator that calculates the expected cumulative shared reward for a specific pair of policies under a game. Mathematically:

$$\rho(\mathcal{G}, \pi_A, \pi_B) = \mathbb{E}_{g \sim \mathcal{G}} \left(\sum_{(s, a_{joint}, s') \in \mathcal{C}(g, \pi_A, \pi_B)} \mathcal{R}(s, a_{joint}) \right) \tag{2.1}$$

Both players seek to maximize ρ, the expected cumulative shared reward.

2.2 Best Response

In a collaborative game $\mathcal{G}$, the best response (BR) to a policy π_x is defined as the policy that maximizes the expected cumulative shared reward given a fixed policy for the other player:

$$BR(\pi_x) = \underset{\pi \in \Pi}{\text{argmax}}(\rho(\mathcal{G}, \pi, \pi_x)) \tag{2.2}$$

2.3 Multi-Task Setting: The Test Bed for Generalization

A learning setting is considered multi-task if the overarching game that the agent is learning in can be decomposed into disjoint sub-games. Mathematically, $\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_{init})$ is a multi-task learning setting with M tasks $\mathcal{G}_i = (\mathcal{S}_i, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_{init,i})$ for $i = 1, ..., M$ if

$$\begin{gathered} \mathcal{S} = \bigcup_{i}^{M} \mathcal{S}_i \\ \mathcal{T}(s^a | s^b, a_{joint}) = 0 \quad \forall a_{joint} \in \mathcal{A} \times \mathcal{A} \quad \forall s^a \in \mathcal{S}_i, s^b \in \mathcal{S}_j \quad \forall \mathcal{S}_i \neq \mathcal{S}_j \end{gathered} \tag{2.3}$$

With this mathematical definition, it is not hard to see why multi-task is the ideal setup to evaluate generalization: once an agent is initialized in a particular task, it is impossible for it to enter states in a different task, making it possible to completely separate training tasks and test tasks.

Chapter 3

The Problem of Human Model Multi-Task Generalization

In this chapter, we define the problem we aim to solve, explain why it is important, and why it is challenging.

3.1 An Introduction to the Human Model Generalization Problem

Consider the following scenario: you are playing a collaborative game with a stranger: first, a random integer between 1 and 10^{12} is selected, and then the two of you vote on whether that number is a prime. Here is how the rewards are distributed:

- +100 if the number is prime and both of you voted "prime"
- +21 if the number is not prime, and both of you voted "not prime"
- +0 otherwise

The best way to learn a good strategy is not immediately obvious, but most will agree that the process involves figuring out *how humans play this game*, since very few people memorize all prime numbers up to 10^{12}.

But now the question becomes, how do we learn about human behavior? Well, let's pair random people up and *collect some human-human data*! We will expect to see that almost all people recognize all even numbers are divisible by 2, most recognize that numbers end in 0 and 5 are divisible by 5. These are the more salient *features* that can usually get picked up by a shallow pass of the human-human data, but what about more obscure ones like "numbers whose sum of digits are divisible by 3 are divisible by 3" or "numbers whose alternating sum of digits are divisible by 11 are divisible by 11"? More importantly, if someone classifies

304538913 correctly as not prime (or misclassifies 64305217 as a prime), how do we know what combination of features they used to reach that conclusion?

But why do we bother with learning these features when we can just collect more human-human data? Indeed, if the size of the interval is relatively small (for example, instead of numbers up to 10^{12}, only consider numbers up to 100), one can collect enough data to know exactly how humans will vote on every single number. However, when the game is over a large interval, abstraction and feature learning become a necessary component, and *generalization* is necessary to model human behavior and find the best strategies.

3.2 Problem Statement

Suppose we have a multi-task collaborative game $\mathcal{G}$ given by tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_{init})$ where $\mathcal{G}$ can be partitioned into disjoint tasks $\mathcal{G}_1, \mathcal{G}_2..., \mathcal{G}_M$.

Suppose we can collect human data on $k << M$ tasks. Without loss of generality, let's assume that the k tasks are $\mathcal{G}_1, \mathcal{G}_2..., \mathcal{G}_k$, and the obtained human-human dataset is denoted as $D_{HH,1:k}$.

We are interested in finding a generalized human proxy H_{proxy} that closely resembles the real human H in unseen tasks. Equivalently, want to find $\pi_{H_{proxy}} \in \Pi$ such that

$$\pi_{H_{proxy}}(s) \simeq \pi_H(s) \quad \forall s \in \mathcal{S} \setminus \cap_{i=1}^{k} \mathcal{S}_i \tag{3.1}$$

To contextualize this mathematically, we define an *abstract* distance functional $d : \Pi \times \Pi \to \mathbb{R}$ such that for any pairs of human proxy policies $\pi_{H_{proxy,a}}$ and $\pi_{H_{proxy,b}}$

$$d(\pi_{H_{proxy,a}}, \pi_H) < d(\pi_{H_{proxy,b}}, \pi_H) \Leftrightarrow \pi_{H_{proxy,a}} \succ \pi_{H_{proxy,b}} \tag{3.2}$$

Since the real reason we want to find a good human model is that it helps the best response optimizer solve the right question, and thus improve collaboration performance with the real human, we can define a key property of d with the notion of Best Response in section 2.2:

$$\pi_{H_{proxy,a}} \succ \pi_{H_{proxy,b}} \Leftrightarrow \rho(\mathcal{G}_{k+1:M}, BR(\pi_{H_{proxy,a}}), \pi_H) > \rho(\mathcal{G}_{k+1:M}, BR(\pi_{H_{proxy,b}}), \pi_H) \tag{3.3}$$

Now the optimization objective for any human modeling technique τ (whose resulting space of possible policy is Π_τ) can be mathematically defined as

$$\pi^*_{H_{proxy,\tau}} = \underset{\pi \in \Pi_\tau}{\operatorname{argmax}} \, \rho(\mathcal{G}_{k+1:M}, BR(\pi), \pi_H) = \underset{\pi \in \Pi_\tau}{\operatorname{argmin}} \, d(\pi, \pi_H) \tag{3.4}$$

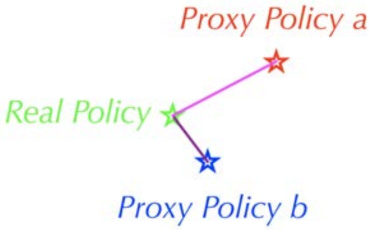


Figure 3.1. Abstract distance function d: the proxy policy b is more preferable than the proxy policy a because $d(\pi_{proxy,b}, \pi_{real}) < d(\pi_{proxy,a}, \pi_{real})$.

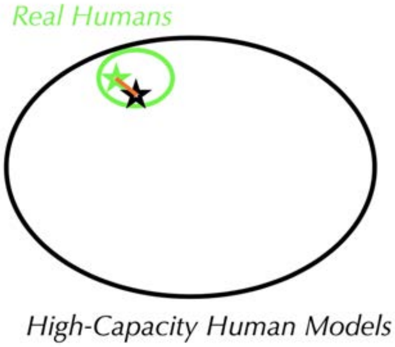


Figure 3.2. Graphical Representation of the objective: given the real human policy π_H (green star) in the space of human policies (green oval), find a solution π (the black star) in the human model space (the black oval) that minimizes $d(\pi, \pi_H)$ (the orange segment).

Chapter 4

Multi-Task Self-Play Initialization as an Inductive Bias

In this chapter, we introduce our approach towards improving generalization to unseen tasks for high capacity human models: initializing behavior cloning agents with multi-task self-play policies.

4.1 The Challenge of Optimizing for Multi-Task Generalization

There is a problem with d, the optimization objective in section 3.2: the true form of this function requires online access to π_H, which makes it unrealistic to optimize over when we are only given limited offline human data. But not all hope is lost, as cross-entropy loss (denoted as CE), the objective function for behavior cloning [12], serves as a good proxy. To formally introduce this:

$$d_{CE}(\pi, \pi_H) = \mathbb{E}_{s \sim \mathcal{S}}(CE(\pi(s), \pi_H(s))) \tag{4.1}$$

Given a human dataset $D^H_{\mathcal{S}} = \{(s^D, a^D)\}$, the empirical objective is:

$$\hat{d}_{CE}(\pi, \pi_H) = \frac{1}{\mid D^H_{\mathcal{S}} \mid} \sum_{(a,s) \in D^H_{\mathcal{S}}} (CE(\pi(s), \mathbb{1}_{\mathcal{A}}(a))) \tag{4.2}$$

Even though $\hat{d}_{CE}(\pi, \pi_H)$ is an unbiased estimator of $d_{CE}(\pi, \pi_H)$ and a convex objective that gradient descent can optimize, the function d_{CE} has one fatal flaw:[1]

$$d_{CE}(\pi_{H_{proxy,a}}, \pi_H) < d_{CE}(\pi_{H_{proxy,b}}, \pi_H) \nRightarrow d(\pi_{H_{proxy,a}}, \pi_H) < d(\pi_{H_{proxy,b}}, \pi_H) \tag{4.3}$$

[1] One example of this is a policy $\bar{\pi}_H$ identical to π_H in all but 1 state in $D^H_{\mathcal{S}}$, but for that state, $\bar{\pi}_H$ predicts a wrong action with 1.0 confidence. This drives $\hat{d}_{CE}(\bar{\pi}_H, \pi_H)$ to ∞, even though $\bar{\pi}_H$ should be highly preferred, yielding extremely small $d(\bar{\pi}_H, \pi_H)$ compared with other policies in the space.

This means that when we use d_{CE} instead of d, the topology of the space is altered, affecting where the optimizer will likely converge to. Figure 4.1 gives a graphical illustration.

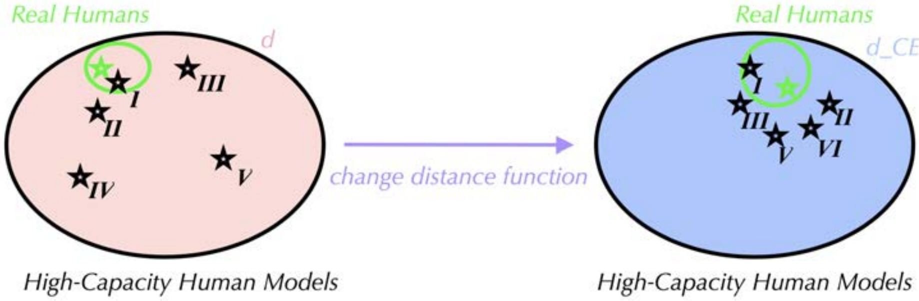


Figure 4.1. When we use cross entropy, a proxy distance function, instead of the real distance function, the topology of the space is changed. Consequently, drastically differently solutions under d ($I \succ II \succ III \succ IV \succ V$) can appear indistinguishable under d_{CE} ($I \sim II \sim III \sim IV \sim V$)

The trouble with solutions of indistinguishable quality mentioned above is compounded by the existence of ambiguous solutions, another common issue with using high-capacity models for human modeling. Even though this issue could be mitigated by collecting more human data (as seen in figure 4.2), the generalization requirements force our human modeling approach to operate in the left side of the spectrum, where there are a huge number of bad solutions that the optimizer needs to dodge.

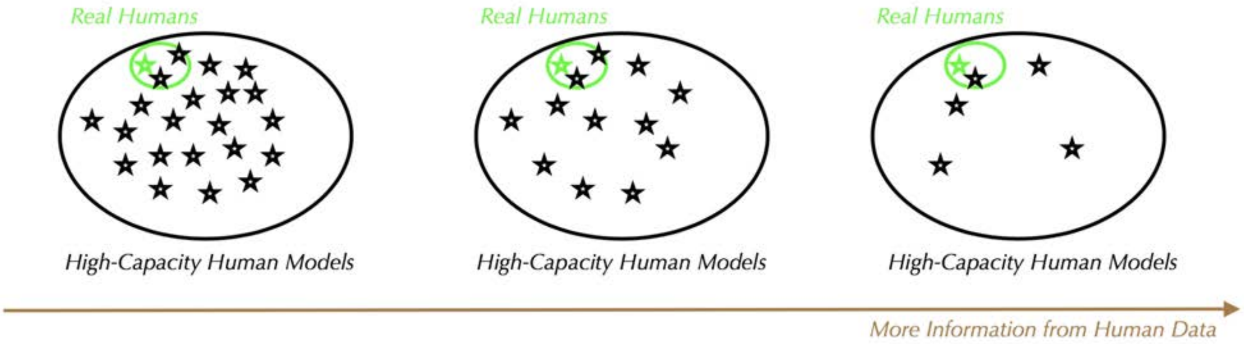

Real Humans
High-Capacity Human Models
Real Humans
High-Capacity Human Models
Real Humans
High-Capacity Human Models
More Information from Human Data

Figure 4.2. More information from human data can help reduce ambiguities among solutions (black stars), and give us better chances at landing near the ground truth (the green star) when the optimizer converges. However, unfortunately the generalization requirements forces us to work in the space towards the left of this spectrum

4.2 Self-Play Policies

Even though a good solution in d appearing as a bad solution under d_{CE} is indeed bad news, it could also be a good news: policies that appear extremely undesirable under d_{CE} (which is why they are normally not considered) could actually be quite desirable under d. Self-play policies fall under this category.

In general, given a game composed of tasks $[\mathcal{S}]$, an optimal self-play policy π^*_{SP} has the following qualities

$$\begin{aligned} \pi^*_{SP} &= BR(\pi^*_{SP}) \\ \pi^*_{SP} &= \operatorname*{argmax}_{\pi \in \Pi} \rho(\mathcal{G}, \pi, \pi) \end{aligned} \tag{4.4}$$

In most coordination situations, self-play policies suffer from a lot of problems: they make too many assumptions about the collaborator, and most importantly, they move and act in very un-human-like manners [4]. In general, if one is to use a self-play policy π_{SP} as a human model, and run counterfactual comparisons on any human trajectory between π_{SP} and an oracle π_H, the expected mean cross entropy loss over the output probabilities is likely to be very high because self-play policies are generally extremely confident about making their actions, which incurs a huge cross entropy loss when the prediction is wrong. Table 4.1 gives an example of this.

π	$\pi(2)$	$\pi(121)$	$\pi(5919)$	$\pi(392471)$	$\hat{d}_{CE}(\pi, \pi_H)$
π_{SP}	(0.9, 0.1)	(0.1, 0.9)	(0.99, 0.01)	(0.55, 0.45)	1.901
π_{random}	(0.5, 0.5)	(0.5, 0.5)	(0.5, 0.5)	(0.5, 0.5)	1.741
π_H (hypothetical)	(0.9, 0.1)	(0.2, 0.8)	(0.02, 0.98)	(0.65, 0.35)	N/A

Table 4.1. Using the prime predictor game from section 3.1 as an example here: every tuple in the table represents $(P(\text{voting prime} \mid \pi, x), P(\text{voting not prime} \mid \pi, x))$. Note that even though π_{SP} is arguably qualitatively better than π_{random}, the one over-confident mistake it makes with 5919 drives up the empirical mean cross entropy loss $\hat{d}_{CE}$ and makes π_{SP} appear a lot worse than how it actually is. In summary, $d(\pi_{SP}, \pi_H) < d(\pi_{random}, \pi_H)$ but $d_{CE}(\pi_{SP}, \pi_H) > d_{CE}(\pi_{random}, \pi_H)$

4.3 Initializing Behavioral Cloning Agents with Self-Play Policies

It is no secret that initialization matters for high-capacity models, especially neural networks, at various levels [7][5]. In this section, we introduce a slightly counter-intuitive initialization technique for high-capacity human models and provide a theoretical justification on why it mitigates the indistinguishable-solutions and ambiguous-solution problem.

To understand why π_{SP} could be desirable, we need to first take a detour to representation learning [2]. Any deep reinforcement learning agent, when learning a policy, implicitly learns latent representations. Theoretically, we can slice any policy into two parts: the representation learner $f : \mathcal{S} \to \mathcal{H}$, and the action learner $g : \mathcal{H} \to \mathcal{A}$. We can write this as a function decomposition:

$$\pi(s) = g(f(s)) = g \circ f(s) \tag{4.5}$$

This may look familiar: if we freeze f, but continue to allow g to update, we have equivalently reduced the capacity of our high-capacity model! You can see the effect in figure 4.3.

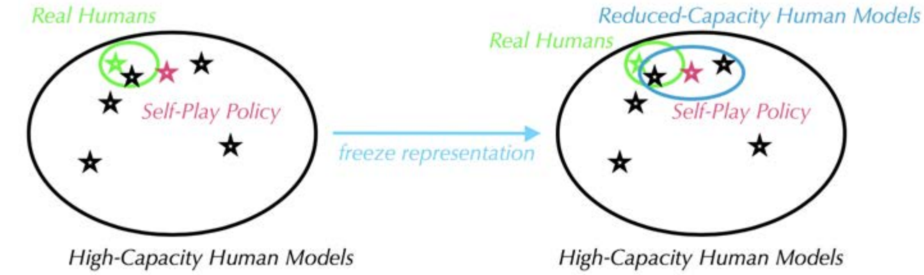

Real Humans
Self-Play Policy
High-Capacity Human Models
freeze representation
Reduced-Capacity Human Models
Real Humans
Self-Play Policy
High-Capacity Human Models

Figure 4.3. By freezing the representation learner, the optimizer is now operating in the space of reduced-capacity models surrounding the self-play policy, and it can continue to improve by updating the action learner. Also note that the constrained region has fewer local optima.

As this point, we see that we can combine the benefits of both high-capacity and low-capacity models. Even though self-play policies output vastly different actions, we can use the learned representation to reduce the effective capacity, which in turn alleviates the ambiguous-solution problem that plagues the high-capacity models.

But what if we go further and allow f to also update? Well that's tricky, because updating f is equivalent to reshaping the constrained region (as seen in figure 4.4, and doing that together with g could possibly put the optimization problem back to the full policy space and reintroduce the ambiguous-solutions problem. However, allowing f to update could be beneficial for picking up latent features that are overlooked by the self-play policies.

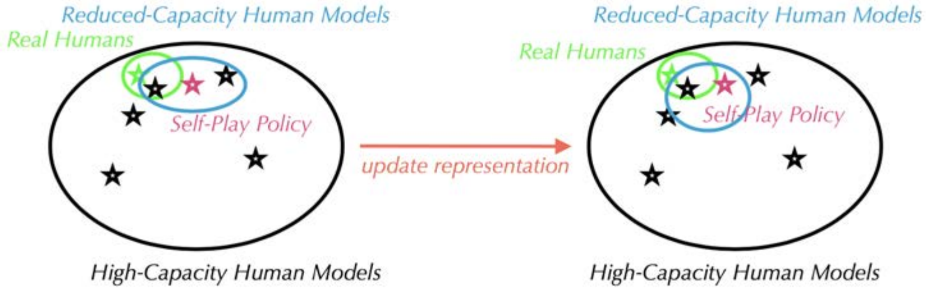

Reduced-Capacity Human Models
Real Humans
Self-Play Policy
High-Capacity Human Models
update representation
Reduced-Capacity Human Models
Real Humans
Self-Play Policy
High-Capacity Human Models

Figure 4.4. Updating the representation learner is equivalent to reshaping the space of reduced-capacity models.

Chapter 5

Experimental Setup

In this chapter, we introduce the evaluation framework, how human data is collected, the human modeling conditions, and the best response conditions

5.1 Framework: Multi-Layout *Overcooked-AI*

Overcooked-AI serves as a great starting point: it was designed to pose challenges on coordination without posing challenges on deep reinforcement learning. But this is not enough, as aside from evaluating methods for human modeling (or AI best responses) for a specific layout, we would also like to see how they generalize to unseen layouts. To evaluate the objective we are interested in, the multi-layout *Overcooked-AI* is constructed by augmenting the *Overcooked-AI* environment with a multi-layout distribution.

The Base Environment: *Overcooked-AI*

Overcooked-AI is inspired from the popular video game Overcooked [6], in which players control cooks in a shared kitchen space to cook and serve dishes according to a set of orders. In *Overcooked-AI*, a simplified version of the game, a team of two cooks are expected to work together to gather ingredients and utensils and coordinate pot operation and route serving in order to accumulate rewards upon successful soup deliveries.

To evaluate the methods, we are particularly interested in the specific setup where one cook is controlled by a human and the other is controlled by an AI.

The Multi-Layout Distribution

To simulate diverse situations in the real world, we define E as a wide distribution of layouts: all layouts that are 7 in size, with approximately 75% empty space and 40% of all counters occupied by an interactive object (onion dispenser, dish dispenser, pot, or serving station). Figure 5.2 illustrates the diversity of layouts in this distribution.

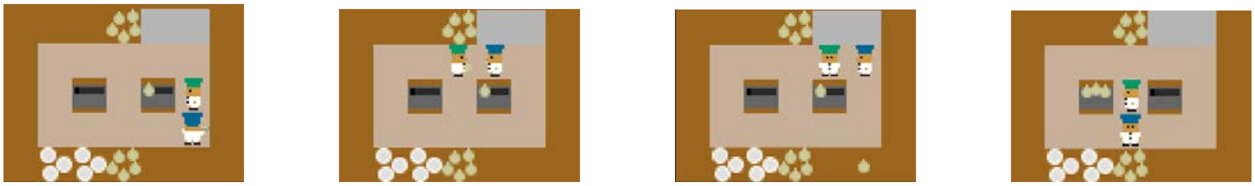


Figure 5.1. Example game-play snapshots from *centre_pots*, one of the fixed layouts in the base *Overcooked-AI* environment

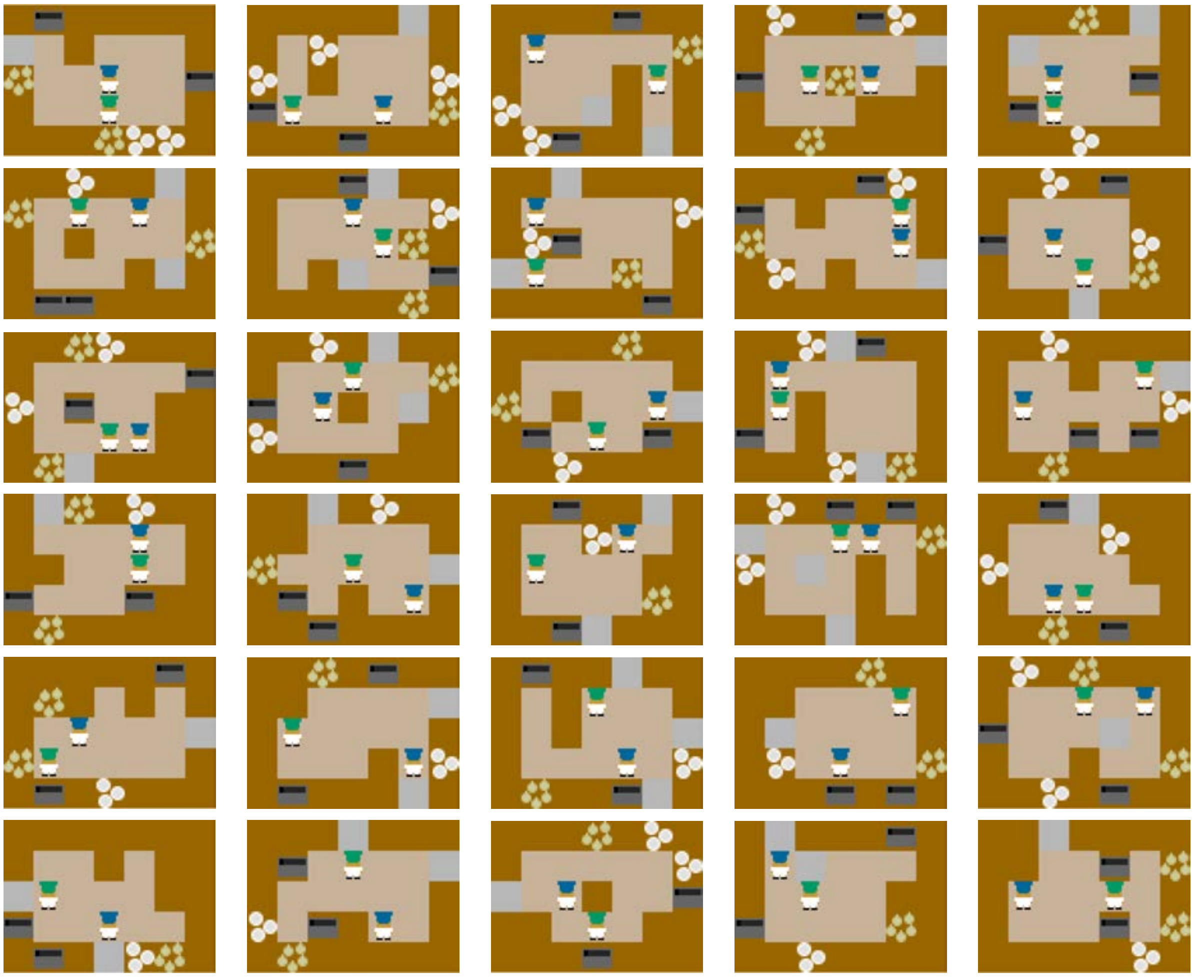


Figure 5.2. 30 out of roughly 10^{13} layouts from the multi-layout distribution E.

The Evaluation Layouts

To evaluate, we sampled 5 layouts from E: e_0, e_1, e_2, e_3, e_4, in increasing levels of difficulties (measured by the the maximum achievable rewards in a single game). These evaluation layouts provides the basis for insights into performance differentials in the general distribution.

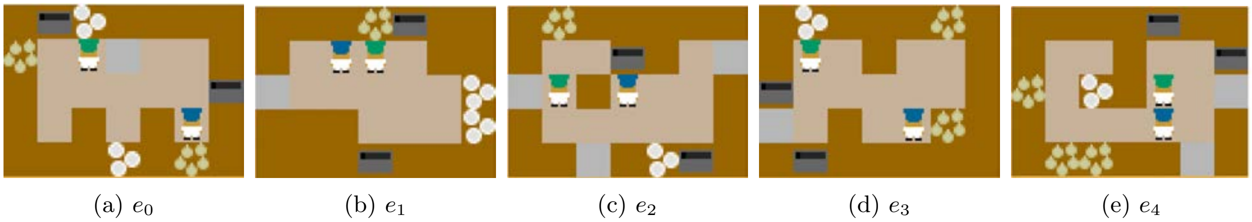

(a) e_0 (b) e_1 (c) e_2 (d) e_3 (e) e_4

Figure 5.3. The five evaluation layouts: e_0, e_1, e_2, e_3, e_4.

5.2 Human Data

Simulated Human

We use a planning-based greedy human model as the ground truth human model H in the experiments to isolate confounding factors (including but not limited to shifts in human behavioral patterns in response to a change in their partner's behavior). A greedy human model re-plans at every time-step and acts noisy-optimally based on an internal set of hidden parameters.

Human-Human Data Collection

To obtain human-human data for human modeling, we divide 40 human participants into pairs $j = 0, 1, 2, ..., 20$. Then each pair j will play 1 game of 400 time steps on each of the 10 layouts:

- 5 evaluation layouts e_0, e_1, e_2, e_3, e_4 (same for all pairs), and
- 5 randomly sampled layouts from $e'_{5j}, e'_{5j+1}, e'_{5j+2}, e'_{5j+3}, e'_{5j+4}$.

In the end, we work with two types of human data:

- Dense and layout-specific human data on all 5 evaluation layouts: $D^H_{e_0}$, $D^H_{e_1}$, $D^H_{e_2}$, $D^H_{e_3}$, $D^H_{e_4}$, and
- Sparse human data across the entire distribution (1 trajectory for each layout): D^H_E.

5.3 Human Modeling: Conditions and Training Setup

All human modeling conditions

For each evaluation layout e_i, the following conditions are considered:

- mlsp: *multi-layout self-play* agent trained on the full multi-layout distribution E only. It is never exposed to human data.

- mlbc: *multi-layout behavior cloning* agent randomly initialized, and only aims to clone D_E^H, the sparse human data across distribution.
- mlbc_mlsp_freeze: *multi-layout behavior cloning* agent initialized with weights from *multi-layout self-play.* The representation learner is frozen, and the agent aims to clone D_E^H by updating the action leaner only.
- mlbc_mlsp: *multi-layout behavior cloning* agent is initialized with weights from *multi-layout self-play.* The agent aims to clone D_E^H by allow updating both the representation learner and the action leaner.
- slsp: *single-layout self-play* agent trained on the specific evaluation layout e_i only. It is never exposed to human data.
- sp: *self-play* similar to slsp, but trained with less attention towards state robustness, so we denote this training distribution $\bar{e}_i$.[1]
- bc: *single-layout behavior cloning* agent randomly initialized, and only aims to clone $D_{e_i}^H$, the dense human data on layout e_i.
- bc_slsp_freeze: *single-layout behavior cloning* agent initialized with weights from *single-layout self-play.* The representation learner is frozen, and the agent aims to clone $D_{e_i}^H$ by updating the action leaner only.
- bc_slsp: *multi-layout behavior cloning* agent initialized with weights from *multi-layout self-play.* The agent aims to clone $D_{e_i}^H$ by updating both the representation learner and the action learner.

Additional Details on Training Human Models

For the overall distribution E, the human data is equally divided into $D_{E,\text{train}}^H$ and $D_{E,\text{validation}}^H$. Similarly, for each evaluation layout e_i, the human data is equally divided into $D_{e_i,\text{train}}^H$ and $D_{e_i,\text{validation}}^H$. All multi-layout BC agents are trained with $D_{E,\text{train}}^H$, and early stopped when the validation loss on $D_{E,\text{validation}}^H$ is minimized. Similarly, all single-layout BC agents are trained with $D_{e_i,\text{train}}^H$, and early stopped when the validation loss on $D_{e_i,\text{validation}}^H$ is minimized.

5.4 Best Responses: Conditions and Training Setup

Before proceeding, let's define PPO_BC[4], a term commonly used in the following discussion.

[1] slsp and mlsp are both trained with diverse-start, a random task initialization method to increase self-play robustness. But sp does not get help from diverse-start.

Method name	Initialization	f updates?	g updates?	Layouts	Human data
mlsp	random	Yes	Yes	E	None
mlbc	random	Yes	Yes	None	D_E^H
mlbc_mlsp_freeze	mlsp	No	Yes	None	D_E^H
mlbc_mlsp	mlsp	Yes	Yes	None	D_E^H
sp	random	Yes	Yes	$\bar{e}_i$	None
slsp	random	Yes	Yes	e_i	None
bc	random	Yes	Yes	None	$D_{e_i}^H$
bc_slsp_freeze	slsp	No	Yes	None	$D_{e_i}^H$
bc_slsp	slsp	Yes	Yes	None	$D_{e_i}^H$

Table 5.1. All human modeling conditions in a nutshell. Note that all single-layout conditions only serve as rough baselines because they have access to dense data directly and are only expected to work on 1 layout, and thus do not need to generalize. The horizontal lines are drawn to separate out 4 groups in this list: all methods in the same group have the same level of information about layouts and human data during training.

When given a human model H_{proxy}, a PPO_BC agent aims to solve the optimization problem:

$$\pi_{\text{PPO\_BC},H_{proxy}} = \underset{\pi \in \Pi}{\text{argmax}}\, \rho(\mathcal{G}, \pi, \pi_{H_{proxy}}) \tag{5.1}$$

All best response conditions

For each evaluation layout e_i, the following conditions are considered:

- mlsp: *multi-layout self-play* agent trained on the full multi-layout distribution E only. It is never paired with a human model.
- mlppobc-mlbc: *multi-layout PPO_BC* agent randomly initialized, paired with mlbc, and trained on the full multi-layout distribution E only.
- mlppobc_mlsp-mlbc: *multi-layout PPO_BC* agent initialized with weights from *multi-layout self-play*, paired with mlbc, and trained on the full multi-layout distribution E only.
- mlppobc_mlsp-mlbc_mlsp_freeze: *multi-layout PPO_BC* agent initialized with weights from *multi-layout self-play*, paired with mlbc_mlsp_freeze, and trained on the full multi-layout distribution E only.
- mlppobc_mlsp-mlbc_mlsp: *multi-layout PPO_BC* agent initialized with weights from *multi-layout self-play*, paired with mlbc_mlsp, and trained on the full multi-layout distribution E only.

- slsp: *single-layout self-play* agent trained on the specific evaluation layout e_i only. It is never exposed to human data.
- sp: *self-play* similar to slsp, but trained with less attention towards state robustness, so we denote this training distribution $\bar{e}_i$.[2]
- ppobc-bc: *single-layout PPO_BC* agent randomly initialized, paired with bc, and trained on the specific evaluation layout e_i only.
- ppobc_slsp-bc: *single-layout PPO_BC* agent initialized with weights from *single-layout self-play*, paired with bc, and trained on the specific evaluation layout e_i only.
- ppobc_slsp-bc_slsp_freeze: *single-layout PPO_BC* agent initialized with weights from *single-layout self-play*, paired with bc_slsp_freeze, and trained on the specific evaluation layout e_i only.
- ppobc_slsp-bc_slsp: *single-layout PPO_BC* agent initialized with weights from *single-layout self-play*, paired with bc_slsp, and trained on the specific evaluation layout e_i only.

Method name	Initialization	H_{proxy}	Layouts	Human data
mlsp	random	mlsp	E	None
mlppobc-mlbc	random	mlbc	E	D_E^H
mlppobc_mlsp-mlbc	mlsp	mlbc	E	D_E^H
mlppobc-mlbc_mlsp_freeze	mlsp	mlbc_mlsp_freeze	E	D_E^H
mlppobc-mlbc_mlsp	mlsp	mlbc_mlsp	E	D_E^H
sp	random	sp	$\bar{e}_i$	None
slsp	random	slsp	e_i	None
ppobc-bc	random	bc	e_i	$D_{e_i}^H$
ppobc_slsp-bc	slsp	bc	e_i	$D_{e_i}^H$
ppobc_slsp-bc_slsp_freeze	slsp	bc_slsp_freeze	e_i	$D_{e_i}^H$
ppobc_slsp-bc_slsp	slsp	bc_slsp	e_i	$D_{e_i}^H$

Table 5.2. All best response conditions in a nutshell. The horizontal lines are drawn to separate out 4 groups in this list: all methods in the same group have the same level of information about layouts and human data during training. Note that one condition is added to group 2 and group 4 to test the importance of initializing best-response with self-play.

[2] slsp and mlsp are both trained with diverse-start, a random task initialization method to increase self-play robustness. But sp does not get help from diverse-start.

Chapter 6

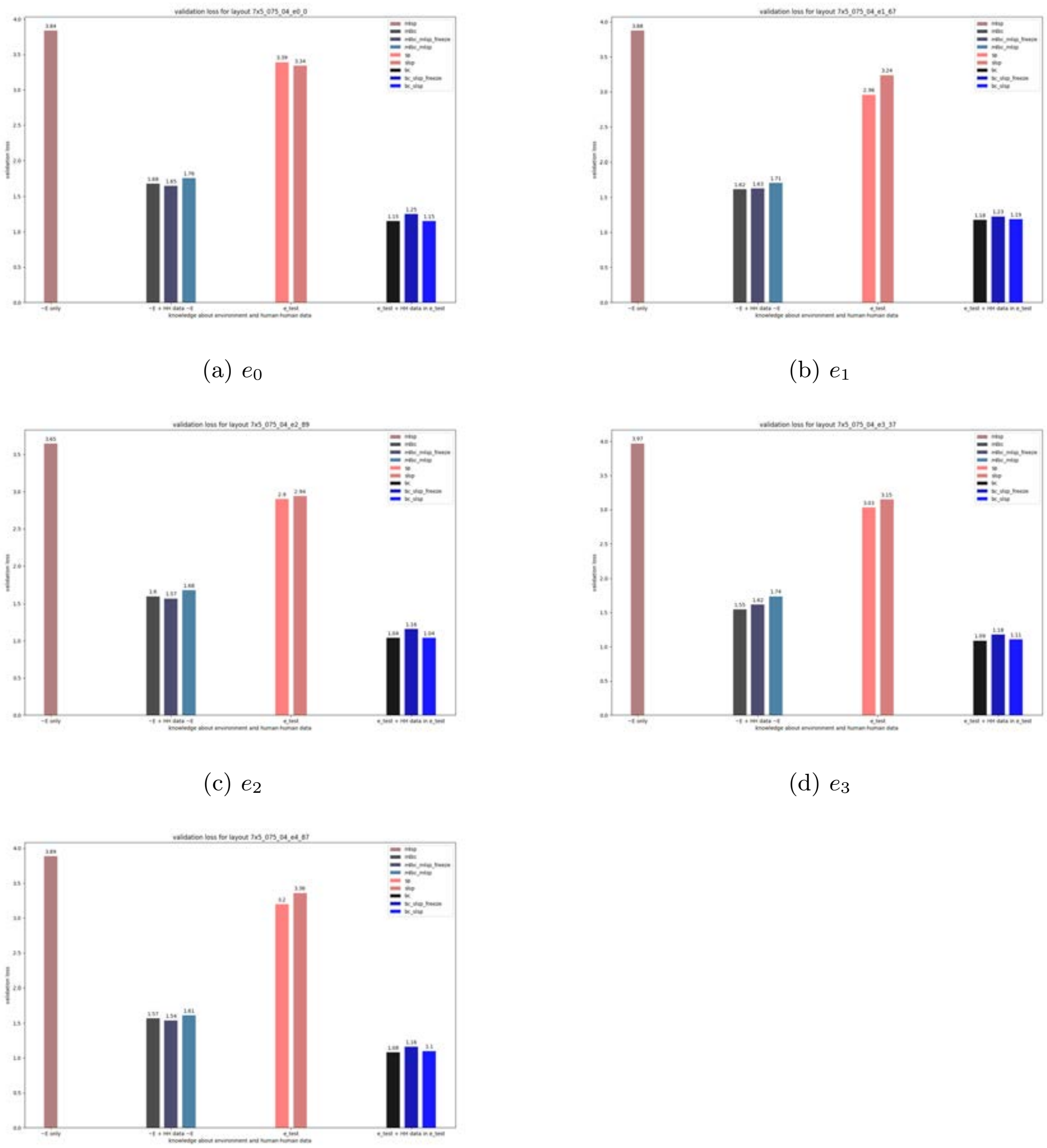
Results

In this chapter, we compare all human modeling conditions using three different distance functions: d_{CE}, d_{reward}, and d.

6.1 The Cross-Entropy Inspired Distance: d_{CE}

First, let's see how the multi-layout methods (which need to zero-shot generalize from sparse off-task human data) stack up with single-layout methods (which have direct access to dense on-task human data), when evaluated on $D^H_{e_i,\text{validation}}$, the validation human data. We plot $\hat{d}_{CE}(\pi, \pi_H)$, the empirical estimate of $d_{CE}(\pi, \pi_H)$ in figure 6.1, and observe a few patterns:

- Self-play policies indeed have huge d_{CE} to the true human policy, regardless of whether they are trained on the full distribution or that specific layout (indicated by big $\hat{d}_{CE}$ for all self-play conditions).
- Behavior cloning can help any type of initialization/freezing reach a point relatively close to the human policy in the d_{CE} space (indicated by similar $\hat{d}_{CE}$ cross conditions within the same group).
- Direct access to dense on-task data allows behavior cloning to reach solutions closer to π_H in the d_{CE} space, if it is an option (indicated by the lower average $\hat{d}_{CE}$ in group 4 compared with group 2).



(a) e_0

(b) e_1

(c) e_2

(d) e_3

(e) e_4

Figure 6.1. $\hat{d}_{CE}(\pi, \pi_H)$ for all conditions on all 5 evaluation layouts. For each layout, the histograms are grouped by the information the condition has about layout and human data. Details about the groupings can be seen in table 5.1.

6.2 The Performance Inspired Distance: d_{reward}

As previously mentioned in section 4.1, even though d_{CE} is great for optimization purposes, it suffers from the fatal flaw of warping the true distance space out of shape. To add another angle to help us pick out the best human model, we again look into the space of distance metrics and ask ourselves: aside from making the human models *act* like a human, what else do we want them to?

One candidate jumps out to us: maybe we also want the human models to *perform* like a human, and the human data has that information too! With the help of the trajectory collector operator $\mathcal{C}$, we can formally introduce the performance-based distance:

$$d_{reward}(\pi, \pi_H) = |\rho(\mathcal{G}_{k+1:M}, \pi, \pi) - \rho(\mathcal{G}_{k+1:M}, \pi_H, \pi_H)| \tag{6.1}$$

Given a human dataset on l unseen validation tasks $\mathcal{G}_{k+1:k+l}$, the empirically performance-based distance is:

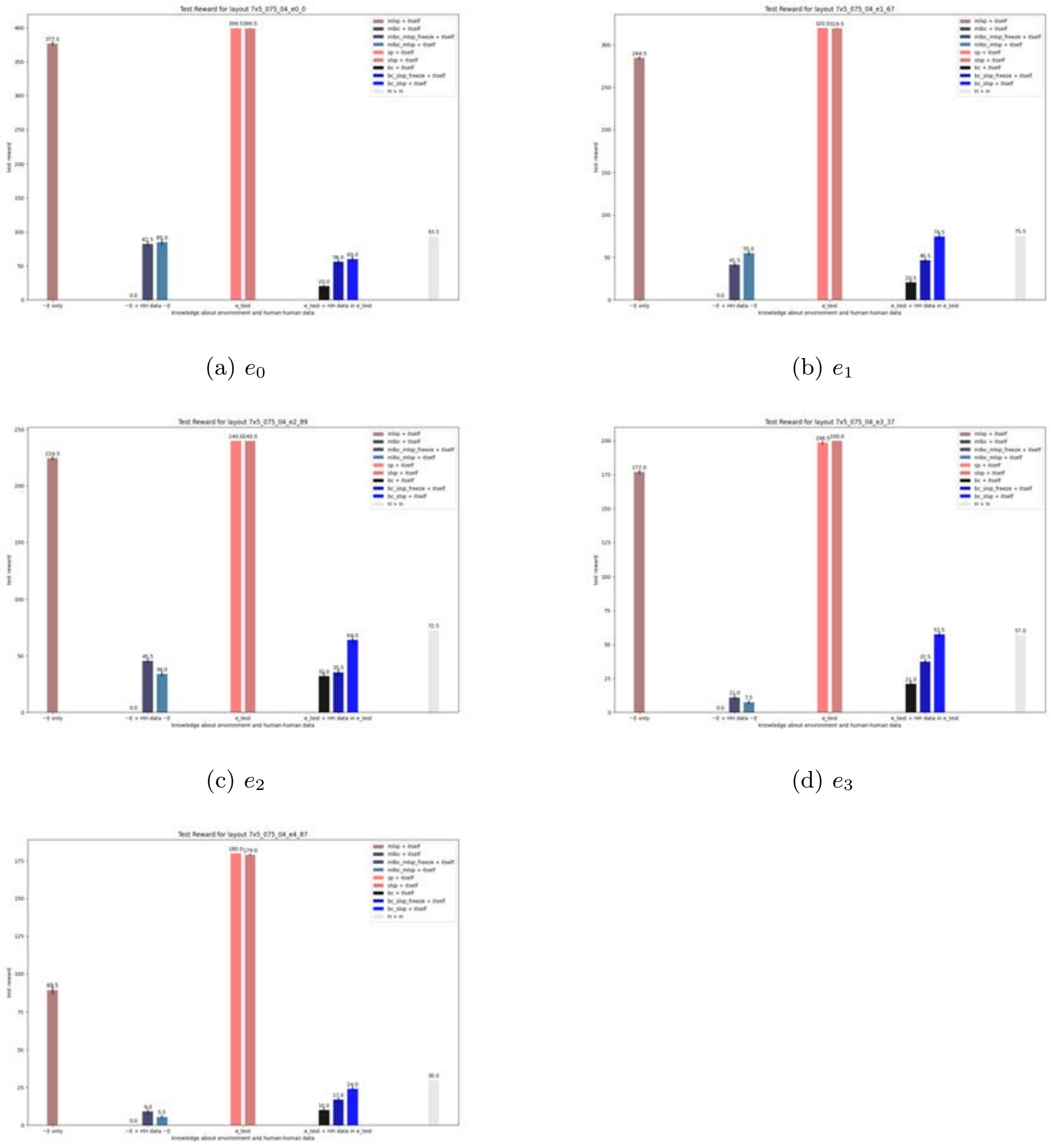
$$\hat{d}_{reward}(\pi, \pi_H) = |\hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi, \pi) - \hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi_H, \pi_H)| \tag{6.2}$$

We plot $\hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi, \pi)$, the intermediate quantity (also known as validation rewards) that will help us get an empirical estimate of $d_{reward}(\pi, \pi_H)$ in figure 6.2 and observe a few patterns:

- Even though the *multi-layout self-play* policy was not trained on any of the evaluation layouts directly, the broad distribution of layouts allow it to achieve similar performance as *single-layout self-play* policies. This indicates that the *multi-layout self-play* policy must have learned a generalizable latent representation that is capable of coping with unseen tasks in the distribution.

- With either slsp or mlsp initialization, behavior cloning also happens to minimize d_{reward}, while not directly performing gradient updates on this particular objective (indicated by the shrinked gap between $\hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi_{trained}, \pi_{trained})$ and $\hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi_H, \pi_H)$).

- With random initialization, behavior cloning performs worse when it has multi-layout sparse data and when it has single-layout dense data. But the random initialization really hurts in the multi-layout case, indicated by the inability of the vanilla mlbc to accumulate any rewards on all 5 evaluation layouts.



(a) e_0

(b) e_1

(c) e_2

(d) e_3

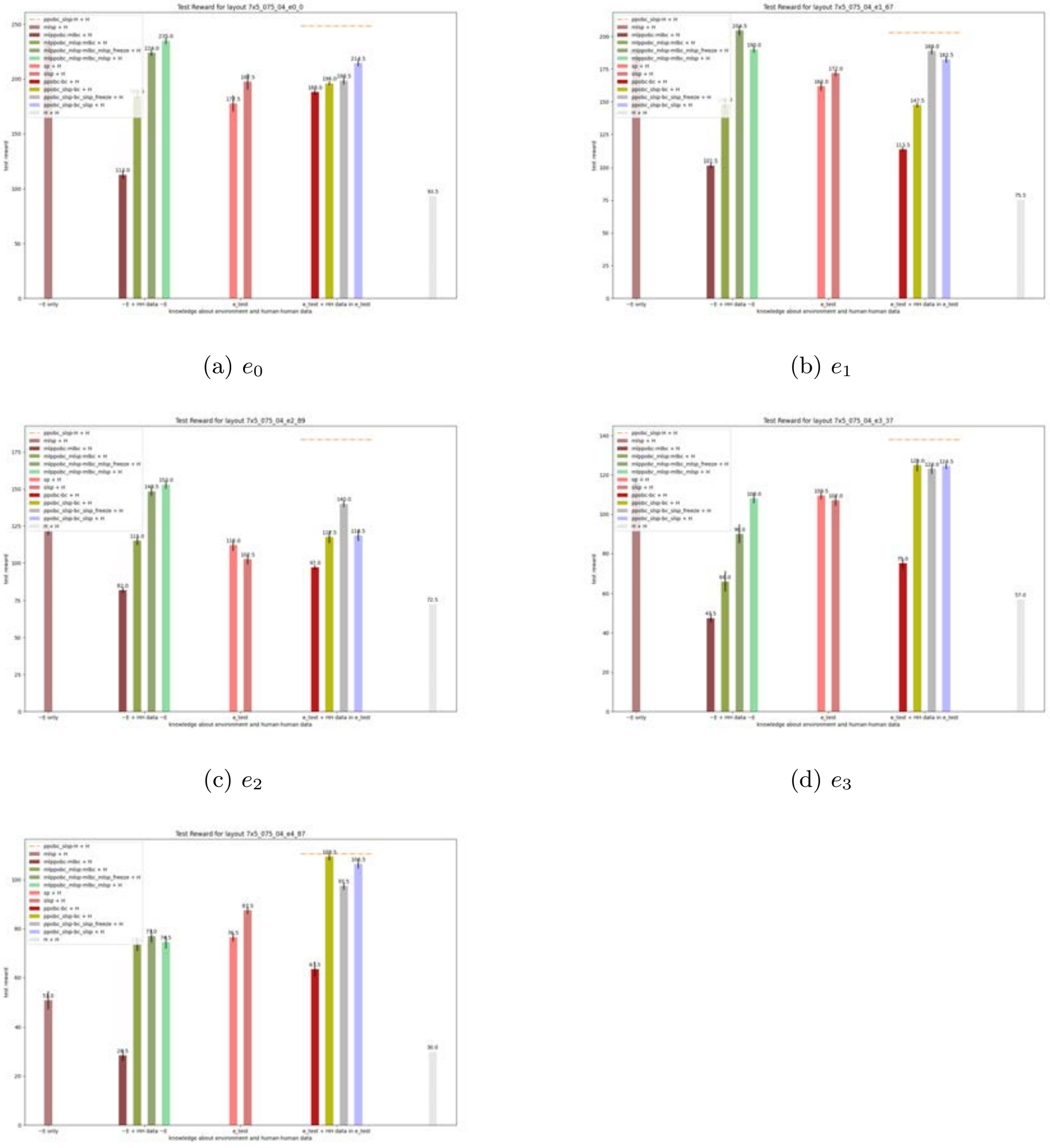
(e) e_4

Figure 6.2. $\hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi, \pi)$ for all conditions on all 5 evaluation layouts. For each layout, the histograms are grouped by the information the condition has about layout and human data. Note that the average human-human reward $\hat{\rho}(\mathcal{G}_{k+1:k+l}, \pi_H, \pi_H)$ is plotted on the rightmost side. Details about the group can be seen in table 5.1.

6.3 The Coordination Rewards with the Ground Truth Human: Information on the Real d

From section 3.2, we know that the true objective for a human model π is to minimize d, which is inversely correlated with $\rho(\mathcal{G}_{k+1:M}, BR(\pi), \pi_H)$. After obtaining the best responses to human models in all conditions, we plot the empirical estimate of this quantity (also known as test rewards) in figure 6.3 and observe a few patterns:

- Even though the self-play policies were never explicitly trained with a human model (because their human proxy is another copy of themselves), they can achieve reasonable test coordination reward.
- Best responses trained with self-play-initialized human models have higher test coordination reward than those trained with a randomly initialized human models, in both multi-layout and single layout situations (indicated by the comparison between the second bar and third & fourth bar in group 2 and group 4).
- The effect of freezing the representation learner f for human models remains inconclusive for now.
- Self-play initialization even improves test reward for the best response policy PPO_BC (indicated by the comparison between the first and second bar in group 2 and group 4).



(a) e_0

(b) e_1

(c) e_2

(d) e_3

(e) e_4

Figure 6.3. Test rewards $\hat{\rho}(\mathcal{G}_{k+1:M}, BR(\pi), \pi_H)$ for best responses learned with the different types of human models. The average human-human reward $\hat{\rho}(\mathcal{G}_{k+1:M}, \pi_H, \pi_H)$ is plotted on the rightmost side. In addition, a golden dotted line is also added to group 4 of each figure: this is the test reward for a PPO_BC agent with direct access to π_H throughout best-response training. Details about the group can be seen in table 5.2.

Chapter 7

Conclusion

Obtaining a human model that can generalize to unseen tasks is an instrumental step towards building an AI system that collaborates well with humans. In this thesis, we formalized the challenge of multi-task generalization in human modeling. We then provided theoretical justifications for why initializing with self-play policies could mitigate ambiguous-solutions and indistinguishable-solutions problems that plague high-capacity human models. Last but not least, we demonstrated the effectiveness of our approach on a collection of sampled layouts in multi-layout *Overcooked-AI*.

There are many future directions one can expand on this thesis. The most promising first step is to see if the results can be replicated with real human data. In addition, it will be worthwhile to attempt self-play initialization for low-capacity models such as Theory of Mind. Last but not least, it will be interesting to see if the results hold in collaborative environments outside the *Overcooked-AI* universe, either on other simulation frameworks or on real robots.

Bibliography

[1] Chris Baker, Joshua Tenenbaum, and Rebecca Saxe. "Goal inference as inverse planning". In: *Proceedings of the 29th Annual Conference of the Cognitive Science Society* (Jan. 2007).

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. "Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives". In: *CoRR* abs/1206.5538 (2012). arXiv: 1206.5538. URL: http://arxiv.org/abs/1206.5538.

[3] Christopher Berner et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. Dec. 2019.

[4] Micah Carroll et al. "On the Utility of Learning about Humans for Human-AI Coordination". In: *arXiv:1910.05789 [cs, stat]* (Jan. 2020). arXiv: 1910.05789. URL: http://arxiv.org/abs/1910.05789 (visited on 10/05/2020).

[5] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Training Pruned Neural Networks". In: *CoRR* abs/1803.03635 (2018). arXiv: 1803.03635. URL: http://arxiv.org/abs/1803.03635.

[6] Ghost Town Games. *Overcooked*. https://store.steampowered.com/app/448510/Overcooked/. 2016.

[7] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*. 2010.

[8] Paul Knott et al. "Evaluating the Robustness of Collaborative Agents". In: *CoRR* abs/2101.05507 (2021). arXiv: 2101.05507. URL: https://arxiv.org/abs/2101.05507.

[9] David V Pynadath and Stacy C Marsella. "PsychSim: Modeling theory of mind with decision-theoretic agents". In: *IJCAI*. Vol. 5. 2005, pp. 1181–1186.

[10] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (2016), pp. 484–503. URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[11] Liting Sun, Xiaogang Jia, and Anca D. Dragan. "On complementing end-to-end human motion predictors with planning". In: *CoRR* abs/2103.05661 (2021). arXiv: 2103.05661. URL: https://arxiv.org/abs/2103.05661.

[12] Faraz Torabi, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation". In: *arXiv preprint arXiv:1805.01954* (2018).

[13] Dizan Vasquez, Billy Okal, and Kai Arras. "Inverse Reinforcement Learning Algorithms and Features for Robot Navigation in Crowds: an experimental comparison". In: *IEEE International Conference on Intelligent Robots and Systems* (Oct. 2014). DOI: 10.1109/IROS.2014.6942731.

[14] Oriol Vinyals et al. *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*.