Towards Building Safe and Robust Human AI Systems

Zhiyang He

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-229
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-229.html

December 20, 2024

Acknowledgement

I am grateful to my advisor, Anca D. Dragan, whose vision and leadership inspired my work in Human-Robot Interaction. Her curiosity and problem-solving skills guided me through complex challenges and kept me motivated.

Thanks to Dylan Hadfield-Menell, Daniel S. Brown, Aditi Raghunathan, Mariah Schrum, and Zackory Erickson for their guidance, and to my co-authors, especially Jeremy Tien and Sashrika Pandey, for their contributions. I also appreciate mentors Akshara Rai, Andrea Bajscy, Andreea Bobu, Micah Carroll, Cassidy Laidlaw, and Eli Bronstein.

My heartfelt thanks to my parents for their support, and to Kechun Liu, who brought joy and meaning to my life.

Towards Building Safe and Robust Human AI Systems

By

Zhiyang He

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Anca D. Dragan
Professor Ken Goldberg
Professor Zackary Erickson

Fall 2024

Towards Building Safe and Robust Human AI Systems

Abstract

Towards Building Safe and Robust Human AI Systems

By

Zhiyang He

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Virtual and physical AI systems are increasingly woven into our daily lives, from self-driving cars and assistive healthcare robots to virtual assistants powered by large language models. However, the dynamic and unpredictable nature of human environments poses significant challenges to their robustness and safety. For example, children or elderly individuals interacting with a house-cleaning robot might inadvertently perform actions that result in hazardous outcomes. Similarly, virtual assistants may exhibit biased responses to certain inputs, such as demographic information, raising concerns about fairness, reliability, and inclusivity. This prompts a critical question: Can these systems operate safely and effectively amidst unpredictable human interactions?

This thesis seeks to address this critical question by drawing on insights from robotics research and applying them to general Human-AI systems. Specifically, it utilizes the reward-rational agent framework—a method commonly used in robotics to predict human actions—and extends it to the design of robust Human-AI systems. The work tackles key vulnerabilities of the framework: (1) reward functions may be mislearned due to causal confusions in the data, and (2) AI policies may be susceptible to exploitation by adversarial human behaviors or misleading contextual information.

To address these challenges, this thesis introduces techniques for active environment synthesis and active human behavior generation, enabling AI systems to anticipate and adapt to unforeseen and edge-case scenarios. Additionally, it explores test-time adaptation to accommodate out-of-distribution users, ensuring greater flexibility in real-world applications. A novel mechanism is also proposed to enable AI systems to controllably focus on relevant aspects of the context, reducing the impact of irrelevant or misleading information. In summary, this thesis tackles the dual challenges of adaptability and robustness, presenting a comprehensive suite of methods to enhance the safety and reliability of Human-AI interactions. It paves the way for future applications where humans and AI agents collaborate seamlessly and effectively to accomplish complex and diverse tasks.

To my mom and dad, Kechun, Didi, Milky and Chickpea

Contents

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Anca D. Dragan. It was Anca's vision and enthusiasm that drew me to the fascinating field of Human-Robot Interaction. Over the years working with her, I have been continuously inspired by her boundless curiosity and remarkable ability to quickly grasp new problems and acquire new skill sets. Her advising style has been a perfect fit for me, as I learn best by observing others. Anca has been an exemplary role model, demonstrating how to identify meaningful problems and remain deeply committed to solving them. From self-driving problems to recommendation systems, to large language models and society. Anca's constant engagement with these challenges has led to a steady stream of intriguing problems for us to tackle — she always thinks deeply about how to make these systems human-compatible, leaving a unique and lasting impact. Our meetings never failed to spark my curiosity and motivate me to explore new ideas. She encouraged me to apply the same research approach to my own interests. While it is common for Ph.D. students completely switch their research statements, I am grateful that this dissertation represents the exact problem I set out to solve when I joined UC Berkeley.

I would also like to extend my heartfelt thanks to the individuals who shaped my research path at Berkeley. Dylan Hadfield-Menell provided invaluable guidance on the intricacies of Inverse Reward Design. Daniel S. Brown challenged me to appreciate the complexities of preference-based learning. Aditi Raghunathan helped me create detailed research roadmaps and introduced me to the world of large language models. Mariah Schrum opened my eyes to the field of Brain-Computer Interfaces, and Zackory Erickson offered keen insights on designing healthcare robots.

I am deeply grateful to my co-authors and collaborators, especially those who contributed directly to the work in this dissertation. Jeremy Tien did outstanding work on assistive gym, and Sashrika Pandey brought incredible energy and creativity to our efforts on engineering large language models.

Special thanks go to those who offered mentorship and invaluable insights during my graduate journey: Akshara Rai, Andrea Bajscy, Andreea Bobu, Micah Carroll, Cassidy Laidlaw, and Eli Bronstein. Your support and guidance have been instrumental.

I am deeply grateful to my parents for their unwavering support throughout my Ph.D. journey. Growing up, I was inspired by my mom's hard work in the chemistry lab, as well as my dad's abilities to handle complex equipments and his resourcefulness in fixing electrical problems at home. Their commitment to learning and problem-solving has been a living example of intellectual curiosity and perseverance, inspiring me to pursue my own academic dreams.

Lastly, I want to thank Kechun Liu, who has brought immeasurable joy and meaning to my life. During the pandemic—an isolating and challenging period, particularly in the first year of my Ph.D. — Kechun was my anchor and my light. We got together, became engaged, and married during these tumultuous years, and he turned my black-and-white world into one full of vibrant colors.

Chapter 1

Introduction

Virtual and physical robotic systems are becoming integral parts our daily lives. As of July 2024, over 700 Waymo cars operate autonomously in the U.S., ChatGPT has more than 200 million users, and autonomous robots are being developed for factory work and elderly care. These systems hold tremendous potential to enhance human well-being and productivity. Research shows that self-driving cars can improve safety and traffic flow, assistive robots could address challenges posed by aging populations, and virtual agents already boost task efficiency by up to 40%. However, fully realizing these benefits requires addressing some critical challenges, namely how to ensure that these systems are safe in the ever increasing complexity of the environments in which they operate.

In fact, autonomous systems have faced high-profile failures in real-world scenarios, raising serious concerns about their safety. In 2023, a Cruise self-driving car caused a pedestrian fatality, sparking public outcry and regulatory scrutiny. Similar incidents, such as collisions involving Tesla's Autopilot, underscore the risks when these systems fail to handle edge cases or unforeseen complexities. These failures often occur in situations where a human operator might have responded more effectively, highlighting the immense challenge of predicting and mitigating errors in increasingly intricate systems. Beyond the immediate risks, such incidents erode public trust and confidence, potentially stalling broader adoption and progress in the industry. As robotics pioneer Rodney Brooks has noted, achieving 99.9% or higher reliability is required for many real-world applications. Doing so, however, is extraordinarily difficult, and reaching every additional "9" requires additional decades of further work.

This is where the dilemma arises: as we aim for robots to perform more and more complex daily tasks—such as cooking meals, loading dishes, or managing household chores—to truly enhance our lives, we face a challenge. To be genuinely useful, these systems must become more capable and versatile. However, our tolerance for safety risks is exceptionally low; even minor errors, such as tipping over hot pans in the presence of family members, are unacceptable. While these ambitious applications have the potential to revolutionize daily life, they magnify the importance of ensuring uncompromising safety. Achieving widespread adoption of autonomous systems will therefore require

breakthroughs not only in capabilities but also in robust safeguards that prevent failures and build public trust. Only by addressing these twin challenges can we unlock the full potential of these transformative technologies.

The goal of my research is to ensure that autonomous agents can be safely and reliably deployed at scale, performing complex tasks in human environments while remaining robust in the face of complicated and unforeseen scenarios. My work focuses on three key pillars of robustness: reward learning, interaction, and adaptation.

Robustness in reward learning is essential for enabling autonomous systems to understand and achieve human goals safely and effectively. For example, a house-cleaning robot must not only rearrange items according to its owner's preferences but also generalize to unfamiliar situations. Achieving this requires learning human preferences from limited data—a challenging task. Our research has shown that existing methods often fail due to causal confusions, leading to unintended outcomes. To address this, we leverage active learning techniques that allow the system to iteratively reduce uncertainties and improve its understanding of human intent, especially in novel scenarios.

Robustness in interactions ensures that robots can operate safely and effectively alongside humans, even when faced with unpredictable behaviors. Humans are not perfectly rational agents and frequently deviate from expected behaviors. For instance, children playing around a household robot might engage in potentially hazardous actions, while elderly individuals may exhibit unexpected movements due to physical limitations. These are rare but realistic scenarios that current systems often treat as "edge cases," handling them in ad hoc ways. For robots to function reliably in real-world environments, they must proactively anticipate and address such edge cases, integrating preemptive safety into their core design rather than relying on reactive measures.

Robustness in adaptation though less immediately visible, is equally critical. Modern adaptive systems, such as online shopping recommendations, news feeds, and social networks, have demonstrated remarkable ability to learn user preferences. However, they often inadvertently pick up on undesirable patterns in human data, amplifying social biases and, in some cases, reinforcing harmful behaviors. To address these challenges, ensuring robust adaptation involves understanding and controlling how these systems interpret contextual information and influence user outcomes. By applying these principles to robotics and virtual assistants, we can create adaptive systems that are not only effective but also equitable and beneficial for diverse users, from children to the elderly. Such robustness allows these technologies to safely and responsibly meet varying user needs while upholding trust and fairness in their design and operation.

By addressing these three dimensions of robustness, my research aims to pave the way for autonomous systems that are safe, trustworthy, and beneficial across a wide range of applications and contexts.

1.1 Thesis Approach

Traditional robotics research model everything outside of the ego agent as the "environment", and the variation of humans and other agents as "disturbances". Similarly, machine learning research treats the training dataset as static and often ignore the dynamic interactions between systems and humans. This becomes limiting for studying human-AI interaction scenarios because human agents can perform purposeful and sometimes arbitrary activities. In extreme cases, such actions can be adversarial: Imagine kids chasing and playing around a house-cleaning robot, or elderly people whose motion deviate from the nominal interactions set by the system designers. These situations can well occur in reality, and not only is it impossible to treat them as the conventional static environment, but it's also difficult to make assumptions about their activities.

Thus, to reason about and facilitate the research on the next-generation Human-AI applications, it is important for us to rethink our designs from system-level. On the one hand, we would like to be able to provide user-level adaptations. On the other hand, we want to push the safety boundary as reliable as possible.

My research aim at enhancing existing algorithms with such use cases in mind, and providing a new framework for us to reason about and quantify their usefulness. More specifically, our approach in human-robot interaction research is to model the human as reward-maximizing agents. We can leverage the human reward function by:

(i) modeling the reward as unknown hidden variables, and by maximizing the information gain, induce assistive behaviours and discover unsafe scenarios;

(ii) modeling and sampling from a distribution of reward functions to simulate human agents and improve the generalizability of AI algorithms,

(iii) modeling the humans as adversarial agents and synthesize unsafe behaviours through optimizing adversarial-natural rewards

This approach is not merely a proposal of a single suite of methods; rather, it introduces a foundational way of thinking about Human-AI interaction that prioritizes both capability and safety. This framework integrates key areas of deep learning research—such as reinforcement learning, learning from human feedback, and deep generative models—creating a versatile platform that can evolve as the field advances. As state-of-the-art methods in generative modeling and optimization continue to improve, they can seamlessly integrate into this framework, enabling us to continually push the boundaries of AI systems in a structured and adaptable way.

1.2 Contributions

The thesis makes the following contributions:

The goal of my thesis is to develop a framework for quantifying and improving the robustness of Human-AI interaction systems.

Learning Robust Human Reward and Preferences

We begin by studying reward learning from humans using preference labels. While recent advancements have demonstrated the feasibility of extending preference learning to deep-learning methods in controlled environments like games, we highlight the significant challenges of applying these methods to more complex, real-world applications, such as assistive robotics. Real-world scenarios often involve intricate and sometimes redundant features, which can complicate the learning process. Preference-based reward learning methods in such settings are prone to errors caused by causal confusion, where the system incorrectly attributes correlations to causation, leading to flawed and potentially harmful reward functions. Through experiments with various learning architectures, we show that this issue is pervasive and demands careful attention in both problem formulation and feature design. Addressing these challenges is essential for ensuring reliable and safe reward learning in practical applications.

To address this issue, we introduce a formalism for actively learning human preferences. We draw inspiration from existing process of reward design, where system designers continually refine reward functions for systems such as self-driving cars to account for new scenarios and failures. The existing passive reward design framework faces two significant challenges: (1) Crafting reward functions in high-dimensional feature spaces is challenging, as it is time-consuming and often leads to redundant solutions that designers may overlook or fail to recognize. (2) Perhaps more importantly, passive reward design is reactive rather than proactive, addressing problems only after they manifest.

To overcome these limitations, we propose a framework called Active Inverse Reward Design (Active IRD). Instead of treating the reward function as fixed, our method infers a posterior distribution over the true underlying reward function. The core innovation of our approach lies in leveraging this posterior to compute reward uncertainty and evaluate the potential information gain of each new design iteration. By strategically selecting new reward design environments to maximize information gain, Active IRD guides designers to focus on the most ambiguous and complex scenarios.

In the context of self-driving car reward functions, our approach systematically identifies scenarios where reward definitions are particularly challenging. This not only reduces reward uncertainty more efficiently than traditional methods but also surfaces novel and informative environments—without requiring explicit pre-specification of the types of scenarios to explore. By actively uncovering and addressing potential gaps in reward design, Active IRD offers a proactive, scalable solution to crafting robust reward functions for real-world systems.

Training with Simulated Humans for Robustness and Adaptation

Humans interacting with robots exhibit diverse movement patterns driven by varying goals and reward functions. To model this diversity, we propose simulating human agents by constructing a distribution of reward functions.

In assistive healthcare robotics, where generating realistic physical interactions is particularly challenging, our framework offers significant advantages over traditional data-driven methods, which require impractically large datasets. By combining reward-based human models with an optimal control framework, we can generate meaningful, robust human actions. Reward functions, being simpler and more interpretable than policy functions, allow us to efficiently create diverse human agents using reinforcement learning. These simulated agents can be used to test robot policies or further train robots.

Building on this synthetic human distribution, we propose a population-based training approach to enable robots to adapt to novel humans. Inspired by test-time adaptation methods in areas like meta-learning and unsupervised representation learning, we introduce a novel learning objective: predicting the next action of the human agent. This objective, akin to next-token prediction in language modeling, is learned alongside robot policies during training and remains fixed during testing. It requires no privileged information about human intentions and movement patterns, instead inferring structure from observed human motions.

Our framework not only learns interpretable representations of human behaviors during training but also enables few-shot adaptation to unseen humans at test time. This leads to faster adaptation to out-of-distribution human actions and improved robustness in human-robot interactions.

Synthesizing Adversarial Humans for Robustness

Robotic and machine learning systems inherently rely on assumptions about the environments in which they operate. This reliance is particularly evident in human-robot interaction systems, where a robot's policy is often trained using a predefined human model or a distribution of models. While system performance can be evaluated within this distribution, it is uncertain how the system will behave when faced with out-of-distribution human actions. For instance, how might a self-driving car react to a sudden, unexpected lane change by another vehicle on a busy highway? Or how would a robot respond if a human partner exhibited unpredictable behavior while collaboratively cutting vegetables? Given the high stakes of such scenarios, addressing unpredictable and rare "long-tail" events has become a major focus for robotics companies.

To tackle this challenge, we propose a framework that combines human reward modeling with adversarial human behavior generation. Starting with a pre-trained robot policy and a set of safety- or progress-oriented reward functions, we synthesize human agents designed to drive the robot toward suboptimal outcomes. These agents, guided by an adversarial reward function, perform actions that intentionally challenge the robot's

policy and expose potential failure points. Notably, this approach is model-agnostic, requiring no access to the internal parameters or architecture of the robot policy, making it broadly applicable.

To ensure the generated adversarial human actions remain plausible, we incorporate a naturalness criterion that constrains the behavior to realistic human actions. By optimizing for both adversarial effectiveness and naturalness, our method produces a range of human behaviors—from fully cooperative to adversarial—forming what we term the "adversarial frontier." This approach automates the discovery of failure cases, uncovering a broader and more diverse spectrum of potential vulnerabilities compared to traditional manual debugging methods.

Ensuring Robust Adaptation via Controllable Contextual Influence

Machine learning systems that adapt to individual user preferences have become the backbone of modern internet applications, driving much of their success and utility. We interact with such systems daily, from personalized news feeds and shopping recommendations to tailored social media experiences. Even the notifications on our phones are finely tuned to align with our preferences, showcasing the ability of these systems to quickly adapt and deliver personalized experiences. However, while these systems offer significant advantages, studies have shown that they can also inadvertently learn undesirable patterns from human data, amplifying social biases and sometimes reinforcing harmful behaviors.

As we advance towards building more powerful robotics applications and large language model (LLM) agents to automate workflows, it becomes increasingly critical to ensure these systems remain robust and safe when adapting to individual users. To address this challenge, we propose a method called Contextual Steering that can be seamlessly applied to foundation models. With Contextual Steering, systems can flexibly adjust their responses in alignment with user-specified directions, which can be defined simply through natural language instructions, without requiring additional supervised fine-tuning. This method also enables us to proactively mitigate and safeguard against potential biases in language generation. We believe that Contextual Steering represents a meaningful step towards systematically enabling robust and equitable adaptations in AI systems, paving the way for safer and more personalized interactions.

Overall, this thesis introduces a framework for modeling human agents via reward functions, which facilitates the generation of diverse human policies, supports the study of personalization and generalization in robot policies, and enables the synthesis of out-of-distribution failure cases.

1.3 Overview

Learning Human Reward Functions

Modeling human as reward-rational agents is useful for understanding and simulating human actions. For that goal, learning a good reward function for human is essential. This section discusses different strategies for learning human preferences, including learning from various forms of feedback and learning through pairwise comparisons.

Learning from Sources of Human Input is an important area of study because it is often difficult to specify reward functions. Researchers have explored a wide variety of different methods to enable robots to learn rewards instead from human input, especially demonstrations [1, 129, 219, 146]. Using this idea, recent works have explored learning from a diverse set of modalities. These include language instructions [198, 50], numerical feedback [13, 88], comparisons [33, 154], advice [131], facial expressions [39], collaborative behavior [63], or corrections [80, 8]. We build on [64], which uses the specified reward function itself as the human input to learn from. We generalize this idea to an iterative, active process for learning reward functions from human

Preference-based Learning refers to the framework of recovering the human reward function by asking for preference over a set of provided queries [199, 33, 12, 22, 133, 165, 109]. It is a popular method for training AI systems when a reward function is unavailable and hard to specify. Preferences are often easier to provide than demonstrations or raw numerical values [199] since they do not require expert proficiency or fine-grained feedback. Leveraging preference comparisons, one can teach robots to learn nuanced human preferences. For example, in an assistive healthcare setting, the robot may ask the patient whether a fast or a slow movement of feeding is preferred. Given that fast movement can be dangerous, and slow movements can cause the food to fall off the spoon, the robot can quickly learn the preferred range of speed of individual patients. In this approach, the system presents two choices to the user and asks them to select the preferred option. By accumulating responses from these comparisons, the robot can infer a preference model that represents the user's likes and dislikes across various parameters.

There are some known challenges to learning preference-based learning. Optimizing a reward function that has been trained using preferences can sometimes lead to unintended behaviors. Anecdotal evidence of this has been documented for Atari games [33, 77, 22], simple robot navigation tasks [82], and language model fine-tuning [170].

Active Learning of Human Preferences involves the robotic system engaging in a structured exploration of human preferences, asking questions or taking actions that elicit responses from users. By doing so, the robot can iteratively refine its understanding of what the human user prefers in various scenarios. For instance, in the self-driving setting, a robot can learn the user or system designer's preference over a set of potentially competing objectives, such as efficiency, comfort and speed. Prior work has focused on actively querying for comparisons [12, 188], demonstrations [23, 42, 41, 110], or reward labels [148]. Such queries are typically for trajectories or states in a given environment (MDP).

Good active learning algorithms should prioritize high-impact questions and adapt based on user feedback, allowing the robot to continually refine its understanding of user preferences over time. Recent work has proposed active learning to synthesizes the environment itself to gain more information about the reward (either the initial configuration and behavior of other agents in the scene, such as [154], or the entire environment [148]). We also propose to actively query new environments, but the human feedback is a proxy reward designed to work only in that environment.

Quantifying Robot Performance on Out-of-Distribution Data

Robotic systems, especially those interacting with complex human environments, face challenges when exposed to out-of-distribution (OOD) data—scenarios or behaviors not covered by training data. Such scenarios often fall within the "long-tail" distribution, where rare, unexpected events can lead to unpredictable behavior or failure. Robots deployed in the open-world need to handle a large set of possible environment configurations, be it a busy road for an autonomous car or a messy kitchen for a home robot. Given this challenge, there has been a rising number of work on safeguarding robot behaviors by generating edge cases. There are a few recent efforts that try to improve robotic performance on OOD data: adversarial examples, domain randomization, edge case synthesis, and robustness in human-robot interactions.

Adversarial Examples in Machine Learning are intentionally modified inputs designed to reveal weaknesses in a model by causing it to make incorrect predictions. The notable examples are in computer vision [178, 56, 97, 37, 27, 38]. Researchers have successfully compromised state-of-the-art computer vision models with simple manipulations of pixels [108, 187] imperceptible to human eyes, which leads to real-world safety concerns. Since then, the community has persistently come up with many attacks [74, 191, 5] and defense mechanisms [169, 35, 37]. Most of the attack methods can be characterized by their allowed perturbation set, such as bounded l2-norm, and the method of optimizing adversarial examples [90]. Robustness methods, on the other hand, often require training against adversarial examples by adding them to the training data [56, 123, 96] as a way of performing robust optimization [164] or focus on regularization and smoothing techniques [60, 35, 145, 37]. Researchers have also found that general adversarial robustness often comes at the cost of performance, due to the elimination of weak correlations in the data [56, 97, 27, 38]. Inspired by prior work in computer vision, our work aims at uncovering adversarial human motions imperceptible from natural human motions.

Natural Adversarial Examples [69, 3, 216] refer to scenarios or inputs that arise organically in the real world and challenge a robot's performance, as opposed to intentionally engineered adversarial inputs. These examples often emerge from unanticipated variations in human behavior, environmental conditions, or complex interactions among multiple factors in the environment. Unlike synthetic adversarial examples, which are crafted to reveal vulnerabilities, natural adversarial examples provide insight into how a system performs under real-world unpredictability. Identifying and testing against these natu-

rally occurring, high-risk scenarios are essential for improving robustness, as they help reveal limitations in perception, decision-making, and adaptability that may not surface during conventional training.

Domain Randomization is a technique that involves training robots on environments with randomized parameters, such as object color, shape, or position, to improve generalization. By diversifying the training data, robots learn to perform tasks across a wide range of conditions, making them more robust to OOD scenarios. This method has proven particularly useful in simulation-based training, where vast amounts of diverse data can be generated efficiently. Domain randomization not only helps robots handle unanticipated conditions but also reduces reliance on exact replications of real-world environments in training, fostering resilience in varied, unpredictable settings. In the past few years, the robotic community have made great progress in ares such as locomotion [180, 206, 95], robotic grasping [112, 81, 176], and deformable object manipulation [200, 162, 70], by leveraging domain randomization [184] for sim2real transfer, where the designer generates a diverse set of static environments to train the robot. By comparison, sim2real transfer for assistive robotics tasks introduces an acting human agent which makes the task more challenging in two aspects. First, it is nontrivial to design "diverse" population of humans to train with — while we can vary the friction of a legged robot in a continuous fashion, it is not as straightforward to generate synthetic humans that span the space of human behaviors, physical limitations, and preferences. Second, since it is difficult to capture the diversity of potential human users, a robot is more likely to encounter out-of-distribution users at test time.

Edge Case Synthesis involves generating rare and challenging scenarios that a robot might encounter in the real world [213], such as unusual human behaviors or extreme environmental conditions. For human-robot interaction, edge cases might include unpredictable human actions, unexpected obstacles, or high-stress situations that test the robot's decision-making and adaptability. [91] and [132] studied using black-box optimizers to perform adaptive testing and generate failure cases. [78] trains a generative model of environments from datasets. [192] generates edge cases by learning a failure predictor. These methods require defining hard constraints that the robot should not violate, which can be incorporated into the reward function via reward shaping [2].

Robust Control in robotics [10] focuses on deriving safety guarantees in the presence of bounded modeling errors and disturbances [168, 208, 209, 158]. It has been applied to human-robot interaction by allowing completely adversarial human motions (the full forward reachable set), as well as a restriction to only motions that are sufficiently likely under a human model [7]. Robustness in reinforcement learning is a growing field inspired by the brittleness of RL policies [211, 53]. Prior work on adversarial attacks focuses on single-agent [140] or two-player competitive settings [75, 114, 124, 196]. In collaborative settings, policies are known to be more brittle [30] than zero-sum games. Recent works have also looked at improving the generalizability in collaborative settings, which is a form of robustness against naturally occurring human distributions [68, 174, 111]. Our work is complementary, providing a way to measure robustness, and perhaps paving the

way to new methods to improve it.

Adversarial Examples in Human-AI interaction. The concept concept of Robustness is essential for Human-AI interaction [213, 153], where small changes in environmental variables—like lighting, object placement, or human behavior—can lead to failures. For instance, in a human-robot collaborative setting, a robot may encounter natural adversarial examples when a human partner suddenly changes their task behavior, like quickly reaching across the workspace or moving in an unusual pattern. In autonomous driving, natural adversarial examples could include rare events like a child darting into the road or unusual lighting conditions at dusk that obscure obstacles. By systematically synthesizing adversarial examples [11], and improving the system on these edge-cases, we can expose robotic systems to challenging, unexpected scenarios, assessing and enhancing their ability to handle OOD inputs. This adversarial testing approach helps developers anticipate failure points and build more resilient models that are less susceptible to subtle, often unforeseen variations in the real world.

Adaptive Autonomous Systems

Recommender Systems are a cornerstone of modern digital ecosystems [151], designed to predict and provide personalized suggestions to users based on their preferences, behaviors, and contextual information. These systems employ a variety of approaches, including collaborative filtering, which analyzes user-item interactions to find patterns among similar users or items; content-based filtering, which leverages item attributes and user profiles to generate recommendations; and hybrid models, which combine multiple techniques to improve accuracy and robustness. They are widely applied in e-commerce for product suggestions, streaming platforms for personalized content delivery, and social networks for curating news feeds and connections. Despite their potentials, these systems often propagate and amplify social biases present in the data they are trained on. For instance, they may inadvertently reinforce echo chambers, exclude minority voices, or promote stereotypical content, perpetuating systemic inequities [121, 28]. Addressing these challenges is crucial to ensuring that recommender systems are not only effective but also fair, transparent, and socially responsible in their deployment [171].

Personalization of Large Language Models. Large language models (LLMs) have emerged as powerful tools for general question answering [25, 186, 83, 59]. One of the main advantage of LLMs over traditional search engines is its ability to customize responses based on individual requests [32]. There have been recently a number of efforts on making LLMs more personalized, by generating natural, context-aware responses, adapting to a user's preferred communication style and providing personalized assistance. These methods commonly rely on retrieval based methods, prompting and finetuning and can be challenging for end-users to quickly prototype with. More recently, with the effort on agentic LLM systems,

Personalization requires LLMs to consider context in a way that improves outcomes for individual end-users. Personalization has been extensively explored in applications

including dialogue agents, movie reviews, and recipe generation [31, 214]. Recent works based on LLM have explored generating more realistic conversational data [194] using dataset of annotated movie dialogues with narrative character personas. Researchers have utilized publicly available reviews and recipe datasets to explore personalization in reviews [103] and recipe generation [113].[201] investigated parameter-efficient models for personalized translation, while [4] have presented a dataset for personalized headline generation derived from real user interactions on Microsoft News.

Bias in LLM Generation [15] highlights issues of bias in language models. The authors investigate how these embeddings often reflect and perpetuate gender stereotypes and introduce an approach to debias word embeddings by identifying a bias subspace. More recent work finds that these concerning biases extend to LLMs. [93] demonstrate that LLMs are three times more likely to choose a stereotype that aligns with a person's gender. Other work has found that LLMs exhibit political bias [126], racial bias [207], and geographical bias [115].

Several approaches have been introduced to counteract bias in LLMs. In their approach, [137] utilized GPT-2 to introduce a substantial reward mechanism aimed at diminishing the occurrence of non-standard outputs. [215] employed data augmentation techniques to substitute gender-specific terms with their antonyms within the initial training dataset, and combined it with another corpus to create a novel model. [84] implemented movement pruning and weight freezing techniques, in addition to employing a debiasing method predicated on a gender-related word projection derived from the work of [86]. The downside to many of these approaches is that they either require modifications to the dataset or extensive model training, both of which are computationally heavy and difficult to deploy.

Chapter 2

Preliminaries

In this section, we discuss the preliminaries required for designing the algorithms in this thesis. We start by introducing the general formulism for Human-AI interaction, with optional hidden variables. We interleave this with research in large language model agents and draw similarities, and discuss the efforts and challenges of learning human prefernces. We then introduce the driving and assistive robot simulation environments, as well as tools of Large Language Models used in our research. We discuss the concurrent research in robotics on sim2real, optimal control and reinforcement learning. We conclude with research on safety and adversarial examples.

2.1 Formalism for Human-AI Interaction Systems

Fully Observable Human-AI Systems

We focus on human-robot systems consisting of an autonomous agent interacting with the human agent in an environment. This can be for instance, an autonomous car interacting with a human-driven vehicle, a healthcare robot arm mounted on a wheelchair feeding the users, or a Large Language Model responding to a user's requests. Our goal is for the autonomous agent to plan in a manner that is cognizant of their effects and interactions with the human [152]. We restrict ourselves to the two agent case in this thesis similar to [152]. We use $\mathcal{R}$ to denote the autonomous agent and $\mathcal{H}$ to denote the human agent.

In the general case, we model the problem as a fully observable dynamical system. Each agents can take an action at a given time, and the action will have consequences on both themselves and the other agent. A state $s \in S$ denotes a continuous state of the system, which include both the states of the human and the robot. At a given point, the human takes an action $a_H \in A_H$ and the robot takes $a_R \in A_R$, they affect the state through a system dynamics $\mathcal{T}$. $\mathcal{T}$ is the transition function where $\mathcal{T}(s^{t+1} \mid s^t, a_R^t, a_H^t)$ is the probability of transitioning from state s^t to state s^{t+1} given $a_R^t \in A_R$ and $a_H^t \in A_H$.

$$s^{t+1} \sim \mathcal{T}(s^{t+1} \mid s^t, a_R^t, a_H^t) \tag{2.1}$$

Assistance as a Two-Player POMDP

In general assistive tasks, there are hidden variables that the robot does not know at the beginning of the task, and need to figure out. For instance, the robot is working with a new human partner and needs to learn about their preference. Following prior work [63, 179, 68] we model an assistive task as a two-agent, finite horizon decentralized partially-observable Markov decision process (Dec-POMDP), defined by the tuple $\langle S, \alpha, A_R, A_H, \mathcal{T}, \Omega_R, \Omega_H, O, R \rangle$. Here S is the state space and A_R and A_H are the human's and the robot's action spaces, respectively. The human and the robot share a real-valued reward function $R : S \times A_R \times A_H \to \mathbb{R}$; however, we assume that the reward function is not fully observed by the robot, i.e., some of the reward function parameters (e.g., the specific goal location or objective of an assistive task) are in the hidden part of the state. , Ω_R and Ω_H are the sets of observations for the robot and human, respectively, and $O : S \times A_R \times A_H \to \Omega_R \times \Omega_H$ represents the observation probabilities. We denote the horizon of the task by T.

Model Predictive Control and Reinforcement Learning

Given a known reward function for the human agent, we can use Model Predictive Control (MPC) to compute the human action. MPC is a popular framework that generates actions given known reward function and environment dynamics in a close-loop fashion at every timestep. Given that computing actions for the infinite horizon is intractible, typically the actions are computed with a discount factor over reward function, which is effective over a finite horizon. Because of this, MPC also known as Receeding Horizon Control. Let $\boldsymbol{s} = (s^1, ..., s^N)$ denote a sequence of states over a finite horizon N, and let $\boldsymbol{a_H} = (a_H^1, ...a_H^N)$ and $\boldsymbol{a_R} = (a_R^1, ...a_R^N)$ denote a sequence of continuous control inputs for the human and the robot. Together the human and the robot receive reward:

$$\boldsymbol{R}(x^0, \boldsymbol{a_H}, \boldsymbol{a_R}) = \sum_{t=0}^{N} R(s^t, a_R^t, a_H^t) \tag{2.2}$$

Given the reward function R, the robot needs to find action sequence a_R so as to maximize the reward over the horizon:

$$\boldsymbol{a_R^*} = \arg\max_{\boldsymbol{a_R}} R(x^0, \boldsymbol{a_R}, \boldsymbol{a_H}) \tag{2.3}$$

.

In some environments where we have access to the simulator, but cannot write the environment dynamics in closed form. For example, this is common for games and robotics

where the environment has complex dynamics. It also applies to interactions with simulated humans, where we have a learned human policy. In these cases, one can apply the framework of Reinforcement Learning, where instead of solving the maximum reward optimization, one can learn a policy that approximates the optimal solution. Among the various Reinforcement Learning algorithms, we apply Proximal Policy Optimization (PPO) [160], which is a form of policy gradient algorithm. We parameterize the robot policy as $\pi_\theta(a|s)$, where θ is the parameter of the policy model, typically implemented as a neural network. The goal is to optimize for θ so as to achieve the best reward, which can be written as

$$J(\theta) = \sum_{s \in S} d^\pi(s) V^\pi(s) = \sum_{s \in S} \sum_{a \in A} \pi_\theta(a|s) Q^\pi(s,a) \tag{2.4}$$

where $d^\pi(s) = \lim_{t \to \infty} P(s_t = s|s_0, \pi_\theta)$ is the stationary distribution of Markov chain for π_θ, approximated using samples in practice. We use $A(s,a) = Q(s,a) - V(s)$ to denote advantage, which is a form of Q-value subtracting the state-value baseline [161], and use $r(\theta) = \pi_\theta(a|s)/\pi_{\theta_{\text{old}}}(a|s)$ to denote the ration between old and the new policy. The objective of PPO is:

$$J^{\text{CLIP}}(\theta) = \mathbb{E}[\min(r(\theta) A_{\theta_{\text{old}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon) A_{\theta_{\text{old}}}(s,a))] \tag{2.5}$$

PPO algorithm has been shown to perform well on robotic benchmarks, and has good scalability to large neural networks compare to previous methods [161]. More recently it has also been applied to optimizing Large Language Models to follow reward functions learned from human feedbacks.

Inverse Reinforcement Learning

The goal of inverse reinforcement learning is to construct a computational model for human actions: the learner agent finds a reward function that explains observations from the human agent performing a set of actions [128, 1]. We model the human as noisily optimizing their own set of reward function. We parametrize the human reward function as a linearly weighted set of features:

$$R_H(x^t, a_R^t, a_H^t) = w \cdot \phi(x^t, a_R^t, a_H^t) \tag{2.6}$$

Here $\phi(x^t, a_R^t, a_H^t)$ is a vector of features, describing different aspects of the task that the human cares about. For instance in the self-driving task, this can include speed, efficiency, comfort, etc. We assume that our feature space sufficiently covers the requirement of the task (typically we can have a large set of redundant features). To infer the weights, one common approach is to apply the principle of maximum entropy to define a probability distribution over possible human actions,

$$P(\boldsymbol{a_H}|x, w) \propto \beta \exp(R_H(x, \boldsymbol{a_R}, \boldsymbol{a_H})) \tag{2.7}$$

This is also called the Bradley-Terry model of human decision making [18] as it characterizes a rationally-optimal human model, where β is a parameter denotes how rational the human agent is.

We can then optimize for w that assigns the highest probability to our observed human actions based on this probability model.

$$w = \arg\max_{w} P(\boldsymbol{a_H}|x, w) \tag{2.8}$$

In practice there are different methods to approximate this optimization. One common approach is to leverage the quadratic approximation trick [102] of the reward function around optimality to simplify Eq. (2.7) which leads to a closed form solution.

Learning from Human Preferences

Learning from human preference is a framework that learns a reward function from preference data. In practice, it is often difficult to ask users to directly label numerical reward values. Instead, it is often easier to provide a set of choices and ask for the preferred option. We build upon the widely used Bradley-Terry model of preferences [18]. Given two possible choices A and B, the Bradley-Terry model estimates the probability that B is preferable to A (denoted $A \prec B$) as

$$P(A \prec B) = \frac{\exp(S(B))}{\exp(S(A)) + \exp(S(B)}. \tag{2.9}$$

where S is a general scoring function that maps from choices to the real numbers. We model the environment as a finite horizon MDP [142], with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, horizon T, and reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The reward function is unobserved and must be learned from preferences over trajectories. Using the Bradley-Terry model [18], the probability a trajectory τ_B is preferred over another trajectory τ_A is given by

$$P(\tau_A \prec \tau_B) = \frac{\exp(r(\tau_B))}{\exp(r(\tau_A)) + \exp(r(\tau_B))}, \tag{2.10}$$

where $r(\tau) = \sum_{(s,a)\in\tau} r(s, a)$ and $\tau = (s_0, a_0, \ldots, s_T, a_T)$.

To learn a reward function from preferences, we assume access to a set of pairwise preference labels $\mathcal{P}$ over trajectories $\tau_1, \ldots, \tau_N$, where $(i, j) \in \mathcal{P}$ implies that $\tau_i \prec \tau_j$. We then optimize a reward function $r_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by θ that maximizes the following likelihood (see Alg. 1):

$$\mathcal{L}(\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{\exp(r_\theta(\tau_j))}{\exp(r_\theta(\tau_i)) + \exp(r_\theta(\tau_j))}. \tag{2.11}$$

2.2 Simulation Environment and Tools

Driving Environment

We adopt the driving simulation model in [155], which treats individual vehicles as point-mass models, and models the cars' dynamics via the physical state $\boldsymbol{s} = [x, y, \phi, v]^\top$, where x, y are the coordinates of the vehicle, ϕ is the heading and v is the velocity. We let $\boldsymbol{u} = [u_1, u_2]^\tau$ represent the control input, where u_1 is the driver's steering input and u_2 is the acceleration. We denote the friction coefficient by μ, and can write the environment dynamics model as

$$[\dot{x}, \dot{y}, \dot{\phi}, \dot{v}] = [v \cdot \cos(\phi), v \cdot \sin(\phi), v \cdot u_1, u_2 - \mu \cdot v] \tag{2.12}$$

This dynamics model apply to all the vehicles (ego and other) in our simulator, and the simulator provides a top-down view of the environment. We extend the simulator to flexibly support a range of environment features that the original environment does not support, such as discrete indicators of collision, as well as different types of cost functions and optimizers. We also re-implement the environment in JAX [17] so that it supports massive parallelization and JIT compilation.

Assistive Gym Environment

The Robotic Caregiving Setup. We leverage the assistive simulation environment in [47], an open-source physical simulation platform that supports a variety of healthcare tasks. More specifically, we focus on feeding and itch-scratching tasks, where there is one robot mounted next to the user, performing the task with the corresponding tools. The human models are equipped with realistic joint models, and the robot observes the keypoint positions of the human body. The overall task reward is defined over a set of features, including penalties for high speed of the robot's end effector, force applied at and outside of the target area, food spillage, etc. The robot policy can be trained with deep reinforcement learning algorithm [160] in less than one day.

We define the observation space for the robot and the human following [47]: the robot observes its own joint angles, and the human's joint positions in the world coordinate and contact forces; the human observes their joint angles (proprioception) and the end-effector position of the robot.

Assistive VR Gym Environment. To support user study of the trained robot policies, we follow [46] and implement our own VR environment where users control the virtual human to interact with the robot in virtual reality. This system gives us the ability to quickly evaluate our trained robot policy over a large number of users, guaranteeing identical environment setups and free us from the hassle of reset and timestep mismatches. Further more, it gives us the ability to study robot failures without the risk of injuries.

In the following figure Fig. 2.1, left and center are the user interacting with virtual robots through the HTC VIVE headset and the hand controller. The right is the first-person view in VR[46].

Figure 2.1: Workstation for performing the VR user study, where we have the human perform interaction with the robot in the calibrated VR environment.

Large Language Models as Interactive Agents

Large Language Models (LLM) are increasingly powerful and widely used for virtual agent applications. The core problem of langauge modeling [143] is unsupervised distribution of estimation from a dataset of sequence examples, each of various length, usually denoted as symbols $(x_1, x_2, ..., x_n)$. Since language has a natural ordering, it is common to factorize the joint probabilities over symbols as the product of conditional probabilities:

$$p(x) = \Pi_{i=1}^{n} p(x_n | x_1, ..., x_{n-1}) \tag{2.13}$$

There have been significant improvements in the expressiveness of modeling choices with self-attention architectures like Transformer [193]. Such models take in a sequence of tokens as inputs, and output the log probability of the next output token in autoregressive fashions. Studies have showed that when trained on internet-scale datasets, the resulting language models are capable of learning new tasks with simple natural language descriptions in few-shot or zero-shot manners.

2.3 Optimizations for Edge-Cases

Reward Design Problem in Robotics

Let an "environment" M be a Markov Decision Process without the reward function. At development time, we assume access to a large set of such environments, $\mathcal{M}_{\text{devel}}$. We assume this is a large set that the designer cannot exhaustively test. This can be, for instance, all the environments in a several-million-mile autonomous driving dataset. Or,

it can be a parameteric set of possible environment configurations, such as placements of objects, roads, and other agents. Finally, it can also be a generative model of the world. The robot will run at deployment time in a different set of environments $\mathcal{M}_{\text{deploy}}$, which we do *not* have access to during development time. [1]

We further assume access to a space of reward functions parameterized by $w \in W$ — these can be linear weights on pre-defined features or weights in a neural network mapping raw input to scalar reward. We denote by w^* weights of the ground truth reward function, which induces the desired behavior in $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$. We denote by $\xi_{w,M}$ the trajectory (or policy, but for this work, we consider deterministic MDPs with set initial states) optimal to the cumulative reward induced by w in M, $\xi_{w,M} = \arg\max_{\xi} R_w(\xi; M)$. Our assumption is that there exists a w^* such that $\xi_{w^*,M}$ is the desired behavior for any $M \in \mathcal{M}_{\text{devel}}$ and any $M \in \mathcal{M}_{\text{deploy}}$. We do *not* have access to w^*.

The designer takes a subset of environments $\mathcal{M} \subseteq \mathcal{M}_{\text{devel}}$ and specify a reward function (via parameters $\tilde{w}$) that leads to good behavior in those environments. In a one-shot design, they would select a set $\mathcal{M}$, assume it is representative of $\mathcal{M}_{\text{deploy}}$, design a $\tilde{w}$ for that $\mathcal{M}$, and deploy the system with $\tilde{w}$ (never changing the reward again). However, in reality, this is an iterative process. They start with an $\mathcal{M}_0$, design a $\tilde{w}_0$, and eventually encounter during their testing a new environment M' on which optimizing $\tilde{w}_0$ does not lead to desirable behavior. Then, they would augment their set to $\mathcal{M}_1 = \mathcal{M}_0 \cup \{M'\}$, and re-design the reward: $\mathcal{M}_0 \rightarrow \tilde{w}_0 \rightarrow \mathcal{M}_1 \rightarrow \tilde{w}_1 \rightarrow ....$ This might continue into deployment if the reward at the time of deployment still fails on $\mathcal{M}_{\text{deploy}}$. Implicitly, at every step along the way, the robot treats the current $\tilde{w}_i$ as equivalent to w^*.

Generative Adversarial Networks

Generative Adversarial Networks (GAN) aims at optimizing for data samples x such that it minimizes [55] the original objective, such as classification or detection. In the original paper, the authors can generate perturbations inperceptible to humans eyes that causes the image to be wrongly classified to other classes. Let D be a discriminator and G be a generator, the objective of GAN can be written as:

$$\max_D \min_G \mathbb{E}_x[\log D(x) + \log(1 - D(G(x))] \tag{2.14}$$

In the context of Human-AI interaction, we can draw the same analogy: perturbations to data points refer to the perturbation in human or robot trajectories. We can define an adversarial human policy $\tilde{\pi}_H$ with respect to an assistive robot policy. In order for the perturbed trajectories to be realistic, we add additional constraints that they are close to the original trajectories. Borrowing from [90, 27], the policy $\tilde{\pi}_H$ minimizes the overall performance under the constraint that $\tilde{\pi}_H$ is similar to the original policy π_H as measured

[1] Our method works best when we can induce a vast set $\mathcal{M}_{\text{devel}}$ that includes all possible scenarios in $\mathcal{M}_{\text{deploy}}$. Otherwise, if $\mathcal{M}_{\text{deploy}}$ is allowed to be drastically different, and if $\mathcal{M}_{\text{devel}}$ is not enough to design a robust reward function, this exposes the robot to unanticipated failures after deployment.

by an f-divergence measure $D_f(\tilde{\pi}_H||\pi_H) \leq \delta$, and adjustable coefficient δ controls the allowable perturbation set in which the adversarial human policy $\tilde{\pi}_H$ deviates from π_H, similar to adversarial perturbations in computer vision [37]:

$$\tilde{\pi}_H(\pi_{R(H)}, d) = \arg\min_{\pi'_H}[R(\pi_{R(H)}, \pi'_H)] \quad \text{s.t. } D_f(\pi'_H||\pi_H) \leq \delta \tag{2.15}$$

Measures such as KL divergence [161, 76, 99, 183], χ^2 [116] divergence are commonly used to characterize the distance between two probability distributions. In the context of policy learning, this can be achieved by training a discriminator $\mathcal{D} : \tau \to \mathbb{R}$ to distinguish between trajectories sampled from $\tilde{\pi}_H$ and π_H. More specifically, $\mathcal{D}$ assigns a score $\mathcal{D}(\approx) := \mathbb{E}_{\tau\sim\pi}[\mathcal{D}(\tau)]$ to any policy π. It is trained to assign low scores to trajectories drawn from the true human policy π_H and high scores to trajectories drawn from the adversarial policy $\tilde{\pi}_H$. We choose the LS-GAN objective [116] for training $\mathcal{D}$:

$$\mathcal{D} = \arg\min_{\mathcal{D}} \ \mathbb{E}_{\tau\sim\tilde{\pi}_H}\left[(\mathcal{D}(\tau)-1)^2\right] + \mathbb{E}_{\tau\sim\pi_H}\left[(\exp\{\mathcal{D}(\tau)+1)^2\right] \tag{2.16}$$

Previous work [116] has proven that this represents χ^2 divergence when trained to optimality: $D_{\chi^2}(\tilde{\pi}_H||\pi_H) = \mathbb{E}_{\tau\sim\tilde{\pi}_H}[\mathcal{D}(\tau)^2]$. In our work, we focous on χ^2 divergence in this paper, in practice one may use other divergence measures such as KL divergence and such as MMD (Maximum Mean Discrepancy) [58]. It can be proved that constrained optimizations under such measures are equivalent to the GAN training objective.

Part I

Robustness in Reward Learning

Chapter 3

Causal Confusion in Reward Learning

In order to model human agents for Human-AI interactions, we need to learn human reward functions. Preference-based reward learning [199, 33, 170, 165] is a popular technique for adapting AI systems to individual preferences and learning specifications for tasks without requiring demonstrations or an explicit reward function. However, anecdotal evidence suggests that these methods are prone to learning rewards that pick up on spurious correlations in the data and miss the true underlying causal structure, especially when learning from limited numbers of preferences [33, 77, 82].

3.1 What is Causal Confusion in Reward Learning?

While the effects of reward misspecification have recently been studied in the context of reinforcement learning agents that optimize a proxy reward function [134], and the effects of causal confusion have been emphasized in behavioral cloning approaches that directly mimic an expert [43, 212, 177], we provide the a systematic study of reward misidentification and causal confusion when learning reward functions.

Consider the assistive feeding task in Fig. 3.2b. Note that all successful robot executions will move the spoon toward the mouth in the area in front of the patient's face—few, if any, executions demonstrate behavior behind the patient's head (the exact likelihood depending on how trajectories for preference queries are generated). In practice, we find that the learned reward will often pick up on the signed difference between the spoon and the mouth rather than the absolute value of the distance. The two correlate on the training data, but using the former instead of the latter leads to the robot thinking that moving behind the patient's head is even better than feeding the person!

Our contribution is a study of causal confusion and reward misidentification as it occurs in preference-based reward learning. First, we demonstrate the failure of preference-based reward learning to produce causal rewards that lead to desirable behavior on three benchmark tasks, even when given large amounts of preference data—in these settings, the learned reward has high test accuracy but leads to poor policies when optimized. We

then study the effect of several factors related to reward model specification and training: the presence of non-causal distractor features, reward model capacity, noise in the stated preferences, and partial state observability. For each of these, we perform an analysis of what errors the learned reward has, how it compares to the ground truth reward, and the amount of distribution shift it induces during policy optimization.

Another of our contributions is to point to the importance of data coverage and interactive techniques that iteratively or actively query for feedback to help learners disambiguate causal features from correlates. Overall, our findings caution that there are many aspects that can make preference learning challenging, from the way we define state—what to include and what *not* to include in the input to the learned model—to how we mitigate the effects of noisy or biased preference data.

3.2 Problem Statement

Environments for Preference Learning. We identify a set of benchmarks that exhibit reward misidentification. In ***Lunar Lander*** [19], the goal is to land a spacecraft successfully. In ***Reacher*** [19] (Fig. 3.2a), the goal is to move an end effector to a desired goal location. In ***Feeding*** [47] (Fig. 3.2b), the goal is to feed the human using a spoon carrying pieces of food. Finally, in ***Itch Scratching*** [47] (Fig. 3.2c), the goal is to scratch an itch location on the human's arm.

Reward Hacking and gaming behaviors are known to commonly occur in reinforcement learning [128, 94] and reward learning [33, 77, 66]. However, these behaviors are often only mentioned anecdotally [94]. Recently, [134] proposed to systematically analyze reward misspecification in RL by creating a set of domains where the agent optimizes a *hand-engineered* proxy reward function and then studying when this leads to incorrect behavior. By contrast, we study cases where the reward function *must be learned*.

Preference Learning is a popular method for training AI systems when a reward function is unavailable [199, 33, 12, 22, 133, 165, 109]. Preferences are often easier to provide than demonstrations or raw numerical values [199] since they do not require expert proficiency or fine-grained feedback. However, optimizing a reward function that has been trained using preferences can sometimes lead to unintended behaviors. Anecdotal evidence of this has been documented for Atari games [33, 77, 22], simple robot navigation tasks [82], and language model fine-tuning [170]; however, there has been no systematic study of causal confusion when learning reward functions.

Causal Confusion in Imitation Learning has previously been studied in the context of behavioral cloning [141, 185]. Prior work shows that behavioral cloning approaches suffer causal confusion due to "causal misidentification," where giving imitation learning policies more information leads to worse performance [43] due to temporally correlated noise in expert actions [177]. Similarly, we find strong evidence of causal misidentification when expert noise is present. [212] use causal diagrams to investigate causal confusion for simplified imitation learning tasks with discrete actions and small numbers of states

when the features available to the demonstrator are different from those of the imitator. By contrast, we study the effects of changes to the observation space when performing preference learning over continuous states and actions and when using non-linear reward function approximation.

Goal Misgeneralization happens when an RL agent has a known goal and some environment features are correlated and predictive of the reward on the training distribution but not out of distribution [100, 163]. This setting is similar to ours in that there is misidentification of the state features that are causal with respect to the reward. However, goal misgeneralization assumes that the ground-truth reward signal is always present during training. However, in contrast to our work, during training the reward signal is always present and always the ground-truth reward. We show that when learning a reward function, the learned reward can be misidentified, leading to poor RL performance. By contrast, [100] and [163] show that even if the learned reward is perfect, RL can still fail due to spurious correlations during training.

Background: Reward Learning from Preferences

We are interested in studying causal confusion that occurs when misaligned reward functions are learned from pairwise preferences over trajectories. As described in Chap. 2, we can model human preferences using the Bradley-Terry model of preferences [18]. Given two trajectories in the environment modeled as a finite horizon MDP [142], with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, horizon T, and reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (unobserved and must be learned from preferences over trajectories). Using the Bradley-Terry model [18], the probability a trajectory τ_B is preferred over another trajectory τ_A is given by

$$P(\tau_A \prec \tau_B) = \frac{\exp(r(\tau_B))}{\exp(r(\tau_A)) + \exp(r(\tau_B))}, \tag{3.1}$$

where $r(\tau) = \sum_{(s,a)\in\tau} r(s, a)$ and $\tau = (s_0, a_0, \ldots, s_T, a_T)$.

To learn a reward function from preferences, we assume access to a set of pairwise preference labels $\mathcal{P}$ over trajectories $\tau_1, \ldots, \tau_N$, where $(i, j) \in \mathcal{P}$ implies that $\tau_i \prec \tau_j$. We then optimize learn a reward function $r_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by θ optimizing the following loss (see Alg. 1):

$$\mathbb{L}(\theta) = -\log \prod_{(i,j)\in\mathcal{P}} \frac{\exp(r_\theta(\tau_j))}{\exp(r_\theta(\tau_i)) + \exp(r_\theta(\tau_j))}. \tag{3.2}$$

In practice, as shown in [21], r_θ can be modeled as deep neural networks that maps environment features to values, and with the preference labels, instead of treating the demonstrations as optimal trajectories, we can recover a function that "explains" their suboptimalities, in return give us the reward function from suboptimal demonstrations. Paired with deep reinforcement learning, we can learn a robot policy that outperforms these demonstrations under the ground truth reward function.

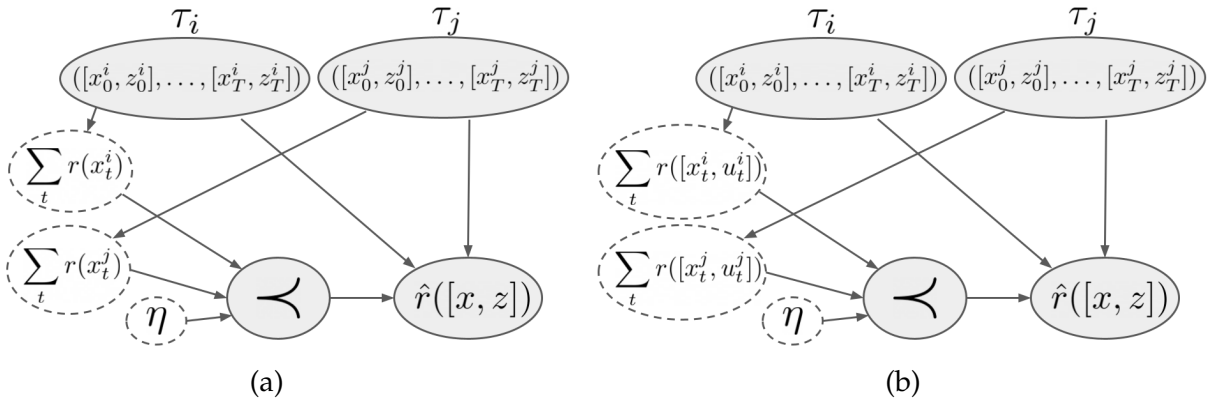


Figure 3.1: **Causal structure of preference-based reward learning.** The reward function $\hat{r}$ is learned from preference labels over trajectory pairs (τ_i, τ_j). Unobserved variables are denoted by dashed lines. Unobservable user noise η affects the preference node. In (a) the true reward is not affected by nuisance variables z. In (b) the true reward is based on unobserved state features u.

3.3 Reward Misidentification

Fig. 3.1 displays the causal structure of preference-based reward learning where pairwise preferences (in the form of a binary label) are given based on an observed reward function r (Fig. 3.1a). There are features x_t^i that are causal and influence $r(x)$, as well as other features z_t^i that are nuisance variables and have no bearing on $r(x)$. Note that z_t^i may very well exhibit correlations with x_t^i, potentially due to their sharing of the same causal parent or there being biases during data collection. Given preference labels over trajectory pairs, the goal of preference-based reward learning is to learn a reward function $\hat{r}(x, z)$ that best matches the stated preferences. In Fig. 3.1b, there are state features u which are causal with respect to the true reward, but are unobserved by the learning agent. In both cases, unobserved human bias and noise, denoted by η, also affects the preferences labels.

The learned reward $\hat{r}$ may be able to achieve low and even near-perfect performance on a held-out test set by making use of 1) nuisance variables that correlate with causal variables or 2) faulty/incomplete correlations between causal variables and r that happen to hold true for the training data distribution. However, performing reinforcement learning on misidentified learned reward values $\hat{r}([x, z])$ leads to distribution shift, resulting in behaviors with low performance under the true reward function r. We define this behavior of learning a reward that achieves low test error but results in poor performance (under the true reward function) when the learned reward function is optimized via RL as *reward misidentification*. We note that the causal graphs we provide in Fig. 3.1 are meant to shed light on a typical way misidentification can occur. In reality, the sources of misidentification can vary widely—a variable may be causal in a certain context and not

in another, causal variables may not be combined properly, etc.

3.4 Experimental Setup

To facilitate reproducibility and encourage future research on causal reward learning, we open-source our code and training datasets. This combination of domains and training data forms the first set of benchmarks for studying reward misidentification and causal reward confusion.

Environments for Preference Learning. We identify a set of benchmarks that exhibit reward misidentification. In ***Reacher*** [19] (Fig. 3.2a), the goal is to move an end effector to a desired goal location. In ***Feeding*** [47] (Fig. 3.2b), the goal is to feed the human using a spoon carrying pieces of food. Finally, in ***Itch Scratching*** [47] (Fig. 3.2c), the goal is to scratch an itch location on the human's arm.

True Rewards and Preference Generation. Each domain has a predefined "true" reward function r. This enables us to create synthetic demonstrations and preference labels via *noise injection*: adding different amounts of noise to an expert policy trained on r (details in Appx. 3.4). As shown by [24], adding this type of disturbance will result in monotonically decreasing performance in expectation and produce a diverse dataset for preference learning. [54] propose a similar approach that switches between an expert policy and a random policy to produce a good coverage distribution over states.

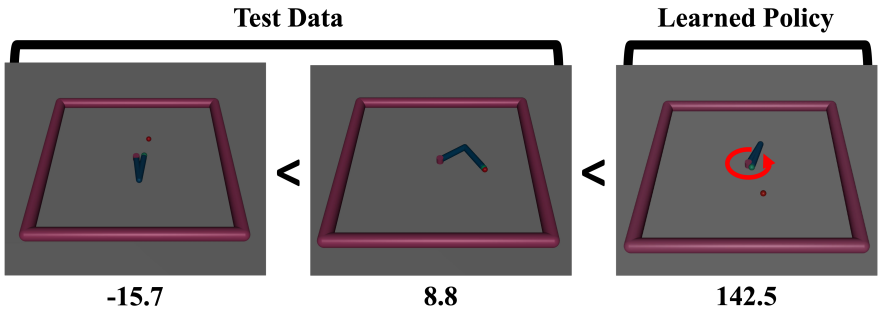
Data Generation Methods

We compare generating trajectories using noise injection versus using different RL checkpoints, as proposed by [21], and find that noise injection leads to similar or better performance. Note that while we use the ground-truth reward function for obtaining preference labels, we assume no access to this reward function during policy learning; we first learn a model r_θ of the true reward function from preference labels $\mathcal{P}$, and then use RL on the learned reward function to produce a policy. We then evaluate the learned policy on the true reward function r.

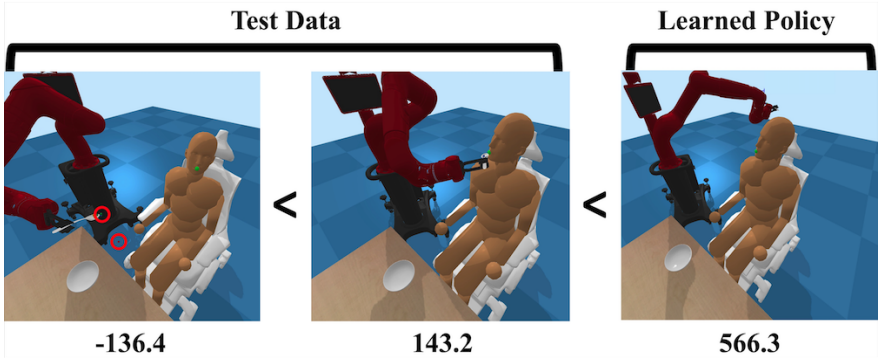
Fig. 3.3 displays a comparison of our method of generating a diverse dataset of trajectory preferences and the method of using a checkpointed RL policy (proposed by [21]).

Ground Truth Reward Functions for Environments

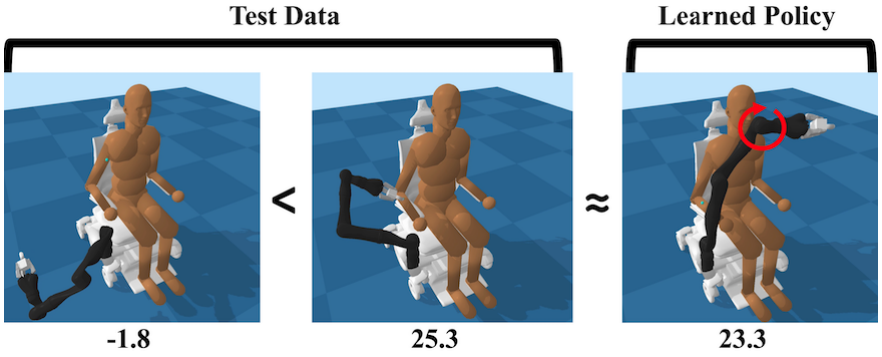
We outline the ground truth reward functions for each environment below. We also refer the reader to the publicly available code repositories for each environment, which describe the rewards in more detail.



(a) Reacher



(b) Feeding



(c) Itch Scratching

Figure 3.2: **Poor behaviors resulting from learned rewards** (rightmost column), despite high accuracy on the test data (left two columns). The predicted trajectory rewards produced by the learned reward function are displayed under each image; each image corresponds to a trajectory.

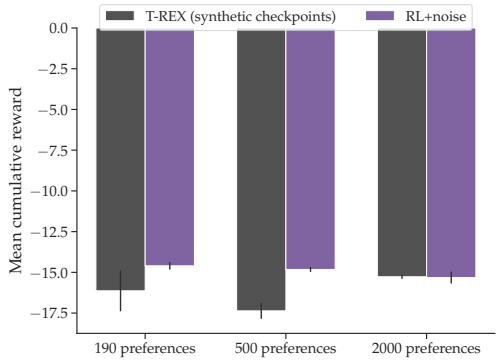


Figure 3.3: **TREX vs. RL+Noise.** Our method of generating diverse trajectories for preference learning performs on par with, if not better than, the method of using rollouts taken from a checkpointed RL policy, as proposed by [21]. Displayed are cumulative trajectory rewards from the Reacher environment.

Lunar Lander (`https://github.com/openai/gym`):

$$\begin{aligned} S(x) &= -100\sqrt{(x^2+y^2)} + -100\sqrt{(v_x^2+v_y^2)} + -100|angle| \\ &\quad + 10(contact_{left}) + 10(contact_{right}) \\ R(x) &= S(x) - S_{prev}(x) \\ R(x) &= R(x) - 0.3(action) \end{aligned}$$

Reacher (`https://github.com/openai/gym`):

$$R(x) = -||position_{fingertip} - position_{target}||_2 + -||action||_2^2$$

Feeding (`https://github.com/Healthcare-Robotics/assistive-gym`):

$$\begin{aligned} r_{distance} &= -||pos_{target} - pos_{spoon}||_2 \\ r_{action} &= -||action||_2 \\ r_{food} &= f(Foods\_in\_mouth, Foods\_on\_floor) \\ \text{pref} &= g(Foods\_hit\_human, Sum\_food\_mouth\_vel, \\ &\quad Spoon\_pos, Prev\_spoon\_pos, Robot\_force\_on\_human) \\ R(x) &= W_{distance} * r_{distance} + W_{action} * r_{action} + W_{food} * r_{food} + \text{pref} \end{aligned}$$

Itch Scratching (`https://github.com/Healthcare-Robotics/assistive-gym`):

$$
\begin{aligned}
r_{distance} &= -||pos_{target} - pos_{spoon}||_2 \\
r_{action} &= -||action||_2 \\
r_{scratch} &= f(Tool\_pos, Target\_pos, Prev\_tool\_pos, \\
&\quad Tool\_force, Prev\_tool\_force) \\
\text{pref} &= g(Spoon\_pos, Prev\_spoon\_pos, \\
&\quad Robot\_force\_on\_human, Tool\_force, Target\_pos) \\
R(x) &= W_{distance} * r_{distance} + W_{action} * r_{action} + W_{scratch} * r_{food} + \text{pref}
\end{aligned}
$$

Evaluating Learned Reward Functions. To establish that a learned reward misidentifies the causal structure, we first check for ***low test error*** and establish that the learned reward performs well on unseen in-distribution test data (thereby ruling out model selection and training failures like not having enough data, capacity, or regularization). We then show that the learned reward fails in two ways: 1) it leads to a policy (called *PREF*) that has *poor performance with respect to the true reward*, and 2) it *prefers its poorly-performing optimized policy PREF over the optimal policy with respect to the true reward (GT)*, indicating that PREF's poor performance is not due to RL failures. Finally, we analyze the learned reward qualitatively via gradient saliency maps and quantitatively via the EPIC pseudometric [54] and KL divergence to elucidate the effects of the reward error by quantifying the distribution shift induced by policy optimization of the learned reward.

Preference Learning Training Details

To learn a reward function from preferences, we assume access to a set of pairwise preference labels $\mathcal{P}$ over trajectories $\tau_1, \ldots, \tau_N$, where $(i, j) \in \mathcal{P}$ implies that $\tau_i \prec \tau_j$. We then optimize a reward function $r_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by θ that maximizes the likelihood:

$$\mathcal{L}(\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{\exp(r_\theta(\tau_j))}{\exp(r_\theta(\tau_i)) + \exp(r_\theta(\tau_j))}. \tag{3.3}$$

This likelihood function is differentiable, allowing us to leverage non-linear function approximation to learn the reward function from trajectory preferences. In practice, we use the Adam optimizer in PyTorch to learn the reward function, r_θ, and then use PPO [160] or SAC [62] for policy optimization given r_θ.

We perform preference learning on three dataset sizes (given in terms of unique pairwise comparisons): SMALL (780), MEDIUM (7140), and LARGE (52326). Our test set is composed of 1770 unique pairwise preferences drawn from a disjoint set of 60 trajectories. See Sec. 3.4 for dataset generation details. Hyperparameters—learning rate and weight decay—are tuned coarsely using the MEDIUM dataset size due to runtime limits and cost of computation. The tuned hyperparameters (best performance on a held-out validation

set) for each environment are as follows: Reacher: weightdecay=0.0001, lr=0.01, Feeding: weightdecay=0.00001, lr=0.001, Itch Scratching: weightdecay=0.001, lr=0.001.

Using the dataset of preferences (SMALL, MEDIUM, or LARGE) obtained offline, we train a neural network reward function approximator with two hidden layers (128 units and 64 units, respectively) and Leaky ReLU activations, after which we perform 1,000,000 timesteps of reinforcement learning with PPO [160] (for Feeding and Itch Scratching) and SAC [62] (for Reacher) using the learned reward function in place of the ground-truth reward function. We optimize the reward function approximator using Adam with weight decay and early-stopping on the validation loss (with a patience of 10 epochs). The full preference-based reward learning procedure we use is detailed in Alg. 1. The hyperparameters for the PPO and SAC agents are as follows (if not specified, then hyperparameters are set to RLLib's default values):

Algorithm 1 Preference-Based Reward Learning

Input: Training dataset of pairwise preferences d_{train}

$\hat{r} \leftarrow$ initialize network (random)
for $epoch \in (0, 100)$ **do**
  for $traj_i, traj_j, label \in d_{train}$ **do**
    $\hat{r}_i \leftarrow \hat{r}(traj_i)$
    $\hat{r}_j \leftarrow \hat{r}(traj_j)$
    $loss \leftarrow CrossEntropyLoss(\hat{r}_i, \hat{r}_j, label)$
    $loss.backward()$
  end for
  $loss_{val} \leftarrow$ calculate cross-entropy loss on validation set
  if $loss_{val}$ doesn't decrease for 10 epochs **then**
    break
  end if
end for
$\hat{\pi} \leftarrow$ Run RL (PPO or SAC) using $\hat{r}$ as reward for 1M iterations

Output: $\hat{r}, \hat{\pi}$

Synthetic Preference Generation

To enhance scalability and reproducibility, we automatically generate a large amount of synthetic trajectory preferences. This was done using an expert RL policy trained using the ground-truth reward function provided with each of environment. We then generate a large number of diverse trajectories by adding ϵ-greedy noise during policy rollouts, where ϵ is the probability that the policy takes an action uniformly at random from its action space. Thus, $\epsilon = 0$ corresponds to the fully trained RL policy and $\epsilon = 1$ corresponds to a uniformly random policy. As noted by [24], adding this type of disturbance will result in monotonically decreasing performance in expectation.

To generate pairwise preferences over trajectories, we select all pairs of trajectories from a set of 40, 120, and 324 total trajectories (for the SMALL, MEDIUM, and LARGE dataset sizes, respectively) generated with ϵ-greedy rollouts for $\epsilon \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We use held-out sets of trajectories for validation and testing. We then use the ground-truth reward functions provided by each environment to provide ground-truth preference labels.

Evaluating Learned Reward Functions

Saliency maps are one of the few methods that allow one to interpret learned reward functions in an isolated, relatively lightweight manner. Following [118], we use raw gradient saliencies, or $\frac{\partial R}{\partial (s,a)}$—the gradient with respect to each element of the input. We extend upon this by examining gradient saliencies *per timestep* along with feature *spread maps*—maps of each input feature's variation (standard deviation, variance, range) over the course of a trajectory.

We produce saliency maps of the learned reward as follows: we forward propagate a single rollout from the learned reward's policy through the reward network. Then, with the reward model's weights fixed, we backpropagate the output from the forward pass through the network and into the input (the policy rollout) to obtain the gradient with respect to each feature in the observation-action pair at each timestep.

EPIC, or Equivalent-Policy Invariant Comparison, is a pseudometric proposed by [54] that quantifies the difference between two reward functions on a given coverage distribution and proposes to be predictive of policy performance without the need for policy optimization. We apply the EPIC pseudometric to compare the distance between various learned rewards and the ground truth reward on the coverage distributions seen during reward learning and those seen during policy training. Using other metrics to compare learned rewards (such as DARD, by [202]) may also prove fruitful in future work.

***Kullback-Leibler divergence*:** We use a discriminator trained to minimize the cross-entropy loss on states from two different distributions following the approach proposed by [76] and [98]. Specifically, we approximate the KL divergence between the distribution of state-action pairs seen during reward learning and those seen during RL on the learned reward. We use this to measure the amount of distribution shift from the reward learning distribution induced by optimizing (potentially misidentified) learned rewards during policy training.

Specifically, for each learned reward, we sample 50 trajectories from the reward's training data and from the resulting policy (taking care to label each trajectory's origin distribution). From these 50x2 trajectories, we create the training and validation splits and then flatten each group of trajectories into a dataset of observation-action pairs. We train a discriminator model (hidden dimensions of 128x128x128) to distinguish between observation-action pairs seen during reward learning and those during RL by minimizing the binary cross-entropy loss. We tune hyperparameters (learning rate, weight decay)

Table 3.1: **Empirical evidence of causal confusion.** We compare policies optimized with a reward learned from preferences (PREF) against policies optimized with the true reward (GT). State features on which preferences are based are fully-observable. Reward functions were trained with 52326 unique pairwise preferences. Both PREF and GT are optimized with 1M RL iterations and averaged over 3 seeds. Despite high pairwise preference classification test accuracy, the policy performance achieved by PREF under the true reward is very low compared with GT. However, *the reward learned from preferences consistently prefers PREF over GT.* This suggests that preference-based reward learning fails to learn a good reward for each of these tasks.

	Pref. Learning Acc.			RL Policy Performance		
Domain	Train	Val	Test	Learned (pref/gt)	True (pref/gt)	Success (pref/gt)
Reacher	0.954	0.956	**0.966**	**44.988** / 3.395	-42.716 / **-5.560**	0.100 / **0.827**
Feeding	0.987	0.976	**0.976**	**277.152** / 124.016	-27.432 / **128.933**	0.603 / **0.990**
Itching	0.954	0.933	**0.928**	**16.588** / 10.282	-47.190 / **248.397**	0.013 / **0.970**

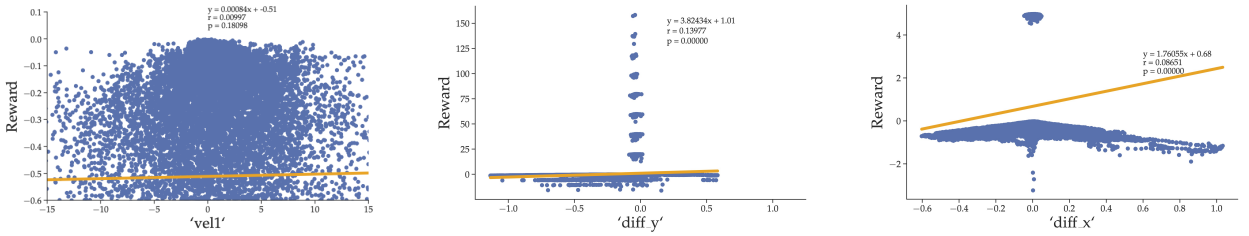
on the validation loss and accuracy.

With this trained discriminator model, we calculate $D_{KL}(p||q)$ by taking the discriminator's negative mean return/logit of all reward learning observation-action pairs, where p is the reward learning distribution and q is the policy optimization distribution. Similarly, we calculate $D_{KL}(q||p)$ by taking the mean return of all RL observation-action pairs. Since the KL divergence is not symmetric, we report $D_{KL}(p||q) + D_{KL}(q||p)$, or the symmetric KL divergence. For a proof on why a discriminator can be used to approximate the KL divergence, we refer the reader to Appendix B.2 of [98].

3.5 Evidence of Causal Confusion

Before varying different factors that affect the performance of the learned reward, we start with a generous setting where we provide large amounts of data and add features to the default observation such that all necessary information needed to infer the ground truth reward, TRUE, is available. We produce the preference training data as detailed in the previous section. Table 3.1 details the results.

We find that the learned reward achieves high preference test accuracies that are comparable to the training accuracy. This indicates that the learned model does not overfit, and there is sufficient model capacity and data. In later sections, we observe that even models with over 99% test accuracy sometimes fail to produce good polices. We also find

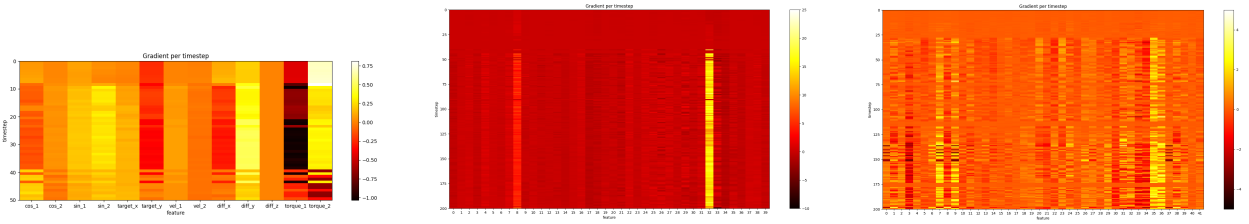


(a) Reacher, 'vel_1' (feature 6)

(b) Feeding, 'diff_y' (feature 8)

(c) Itch Scratching, 'diff_x' (feature 7)

Figure 3.4: **Correlations of Spuriously Weighted Features with Reward.**



(a) Reacher

(b) Feeding

(c) Itch Scratching

Figure 3.5: **Gradient Saliency Maps for Fig. 3.2**. Fig. 3.5a shows the slight positive gradient for the angular velocity feature 'vel_1', which results in spinning behavior. Fig. 3.5b shows the nearly constant positive gradient in the 'diff_y' feature (feature 8) that incentivizes the robot to move into, past, and behind the human's head. A properly learned reward would exhibit a gradient that 'flips' to negative the moment the robot arm goes behind the human's head. Fig. 3.5c shows significant positive gradients for features 7, 9, 35, and 36, which correspond to 'diff_x', 'diff_z', and the last two degrees of freedom in the 7-DOF robot arm, respectively.

that the learned reward *prefers* PREF to GT. This shows that the poor performance is not due to an issue in RL training. Unfortunately, the actual performance of PREF is disastrous: it has poor TRUE reward (compared to GT) and poor success rates. Overall, the learned reward incentivizes poor behavior, despite high test accuracy.

For Reacher, we observe that PREF chooses to simply spin very fast rather than reaching for the target. We note, however, that the learned reward correctly classifies the leftmost image in Fig. 3.2a (where the agent just folds its arm) as being worse than the middle image (where the agent successfully reaches the target), and does so with 96.6% accuracy on such diverse pairs of comparisons. One would think that this would apply to the behavior in the rightmost image, where the agent folds its arm as well (and subsequently spins), but the learned reward turns out to strongly prefer the rightmost case.

For Feeding, the learned reward encourages minimizing the *signed* difference between

the spoon and the mouth rather than the *absolute value* of the difference. This is because the majority of trajectories approach the mouth from in front. Fig. 3.4b shows that there are far more states that have low reward at negative 'diff_y' values than states that have low reward at positive 'diff_y' values. As a result, the learned reward (Fig. 3.5b) correctly identifies spilling food (left) as being worse than feeding (middle), but goes further and incentivizes bringing the spoon towards and even *behind* the head in an attempt to minimize the signed difference. For Itch Scratching, the learned reward correctly identifies flailing (left) as being worse than actual scratching (middle). However, as seen in

Fig. 3.5c, the Itch Scratching agent (spuriously) learns a higher weight on two components of the action (corresponding to two robot arm joints), which results in the agent turning the last section of the arm in a circle while trying to keep the end effector close to the itch target—another type of flailing! It also falls into the same trap as the Feeding agent in minimizing the signed rather than the absolute difference (Fig. 3.4c shows there is a bias toward negative values of 'diff_x' rather than an even distribution of states across both positive and negative values). Fig. 3.2 summarizes this behavior. Appx. 3.5 provides plots of the aforementioned features' spurious correlations with the true reward and gradient saliency maps of the learned rewards.

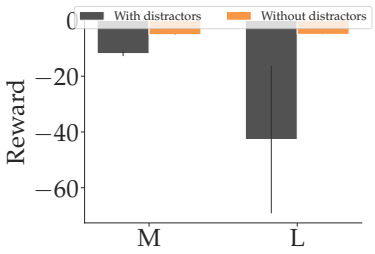
3.6 Factors that May Lead to Causal Reward Confusion

We examine various factors of the preference-based reward learning problem setup and their effects on reward misidentification. The motivation for exploring each is as follows:

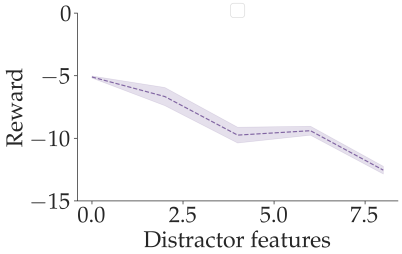
Our experiments with ***Distractor Features*** draw directly from the findings in Sec. 3.5, where we find that certain non-causal features may be spuriously correlated with the reward. Experiments on ***Model Capacity*** are inspired by [134]'s findings on the effects of increased agent capabilities on reward hacking. Exploring ***Noise in Stated Preferences*** is inspired by the fact that preference data collected from humans is often rife with various biases and noise. Studying ***Partial Observability of Causal Features*** is motivated by the fact that it is not always possible to fully-observe all causal features in the real world. ***Complex Causal Features*** is inspired by the observation that the causal reward is often a complex function of state variables and, in complex tasks, features may be causal in some contexts and non-causal in others. The above factors (which may often be overlooked) can contribute significantly to reward misidentification and the eventual success or failure of the reward model.

Distractor Features

One reason for reward misidentification is the presence of nuisance features in the input that are spuriously correlated with preference labels. To study this effect, we incrementally remove such 'distractor' features and test the impact this has on the learned reward. For example, for Reacher, we know that joint angles and angular velocities are not causal



(a) RL Performance



(b) Removing distractors

Dataset Size	Distractors With	Without
M	**53.731**	6.762
L	**46.032**	7.366

(c) KL Divergence

Figure 3.6: **Distractor features.** Fig. 3.6a compares performance with distractors present in the learned reward with performance without distractors present in the learned reward across two dataset sizes. Fig. 3.6b is a sensitivity analysis on the number of distractor features done on the M dataset size case from Fig. 3.6a. In Fig. 3.6c, we see that distractor features result in a much larger distribution shift (as measured by the KL divergence between state-action pair distributions at reward learning and RL).

to the ground truth reward, which is based purely on distance between end effector and target and the norm of the action (a control penalty) (see Sec. 3.6 for a complete list of causal and distractor features). As we see in Fig. 3.8, removing all distractors drastically improves performance, and, indeed, the more such features are left in the input, the worse the performance gets in Reacher, almost linearly. Granted, removing distractors does also slightly improve the validation and test performance in this case—indicating that there is some signal in the training data to help discern the spuriousness of certain features. Nonetheless, over the several experiments in this paper, we find that the validation error is not strongly correlated to performance—in some cases, increasing the validation accuracy results in worse performance.

Input Features for Environments

Features that are distractors are labeled with a *D*. Features that are causal are labeled with a *C*. The dimensions of each feature are included in parentheses.

Lunar Lander:

- Causal Features: x-coordinates of the lander (1), y-coordinates of the lander (1), Linear velocity in x (1), Linear velocity in y (1), Angle of the lander (1), Whether each leg is in contact with ground (2), Action — one of {do nothing, fire left orientation engine, fire main engine, fire right orientation engine} (1)
- Distractor Features: Angular velocity of the lander (1)

Reacher:

- Causal Features: Position_fingertip - position_target (3), Action — Torque applied at the hinges (2)
- Distractor Features: Cos of angle of the arm (2), - Sin of angle of the arm (2), Coordinates of target (2), Angular velocity of the arm (2)

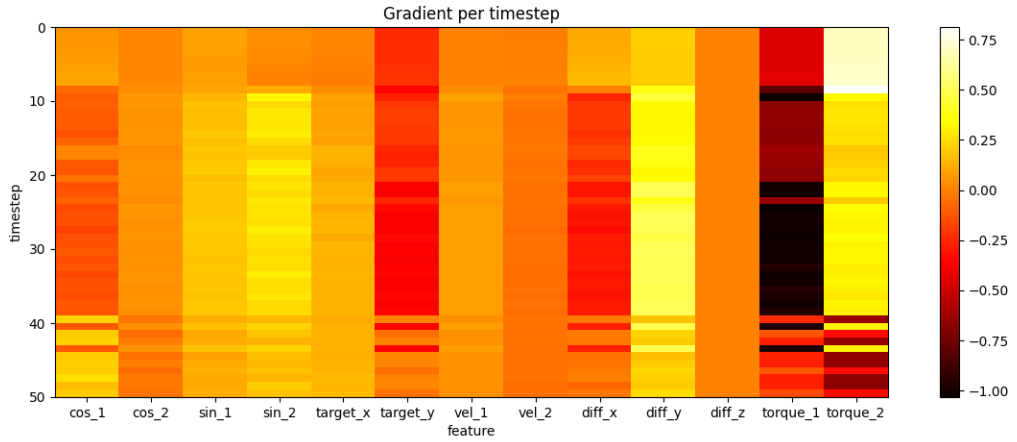
Feeding:

- Causal Features: spoon_pos - target_pos (3), Spoon_force_on_human (1), Action (7), Foods_in_mouth (1), Foods_on_floor (1), Foods_hit_human (1), Sum_food _mouth_vel (1), Prev_spoon_pos (3), Robot_force_on_human (1)
- Distractor Features: - Spoon_pos (3), Spoon_orient (4), Robot_joint_angles (7), Head_pos (3), Head_orient (4)

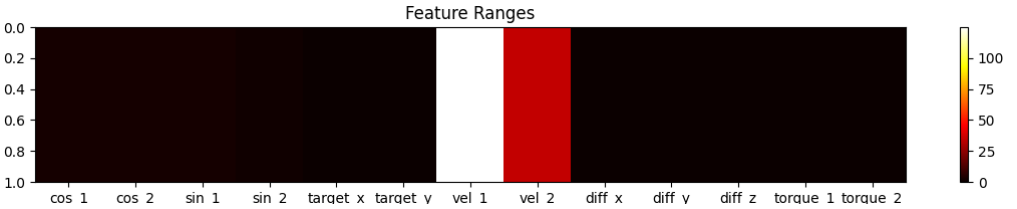
Itch Scratching:

- Causal Features: Tool_pos (3), Tool_pos - target_pos (3), Tool_force (1), Action (7), Prev_tool_pos (3), Robot_force_on_human (1), Prev_tool_force (1)
- Distractor Features: Tool_orient (4), Target_pos (3), Robot_joint_angles (7), Shoulder_pos (3), Elbow_pos (3), Wrist_pos (3)

Fig. 3.2a depicts the policy of the learned reward with distractors—the Reacher robot learns to simply fold its arm and spin. Looking at the gradient saliency maps (Fig. 3.7) illuminates why this is the case. Firstly, in Fig. 3.7a, we observe that the learned reward is misaligned with respect to 'diff_y', the feature corresponding to the difference between end effector position and target position along the y-axis; specifically, we note that the learned reward actually rewards an increase in 'diff_y', rather than a decrease. Next, we notice that the reward doesn't penalize action evenly. It penalizes 'torque_1' (the one responsible for the spinning), but rewards 'torque_2' across nearly all timesteps. It should instead be largely a negative penalty, as seen in Fig. 3.9a. Lastly and perhaps most importantly, we observe in Fig. 3.7b that 'vel_1' and 'vel_2' have very large feature ranges, corresponding to the large variations in angular velocity achieved by the Reacher robot's spinning behavior. Looking back to the gradient for each of these velocity features in Fig. 3.7a, we observe that the reward has a slight positive gradient with respect to each which stems from a slight correlation between angular velocity and reward. Fig. 3.4a shows this correlation; there is a slight bias in the training data toward states with high reward and high velocity. By spinning fast, the Reacher robot is thus able to achieve much higher (learned) reward than performing the proper reaching action. The KL divergence (Table 3.6c) between the distribution of observation-action pairs seen during reward learning and those seen during policy optimization provides further insights—in incentivizing the Reacher robot to spin fast, it leads the RL optimization toward states that were not seen during reward learning (specifically, states where the robot is spinning very fast).



(a) Gradients - With distractors



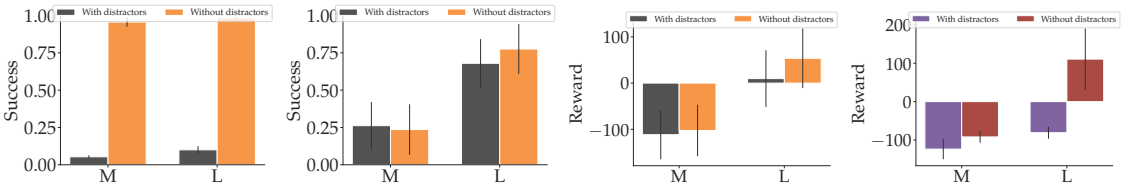
(b) Feature Ranges - With distractors

Figure 3.7: **Saliency maps: distractor features.** Fig. 3.7a is a gradient saliency map of the (misidentified) learned reward. Fig. 3.7b is a heatmap displaying the ranges of each feature over the course of a trajectory produced by the learned reward. The large range of the 'vel_1' feature and its corresponding positive gradient are the reason why the trajectory does well under the learned reward.

Section 3.6 displays results for the other tasks. Similar to the Reacher task, we find that removing distractor features generally helps performance. Note that the reason removing non-causal distractor features does not appear beneficial for Feeding is that the main spurious correlation (discussed in Section 3.5) involves one of the *causal* features—namely, the difference between spoon position and target position. Thus, removing purely non-causal features fails to address this issue in Feeding.

Distractor Features

Fig. 3.8 shows the additional results for removing distractors. Fig. 3.9 shows the saliency map for the Reacher learned reward without distractors. Table 3.2 shows additional KL divergences for the Feeding and Itch Scratching environments. We note here that the reason removing non-causal distractor features appears to not benefit for Feeding is that the main spurious correlation (discussed in Section 3.5) involves one of the *causal*



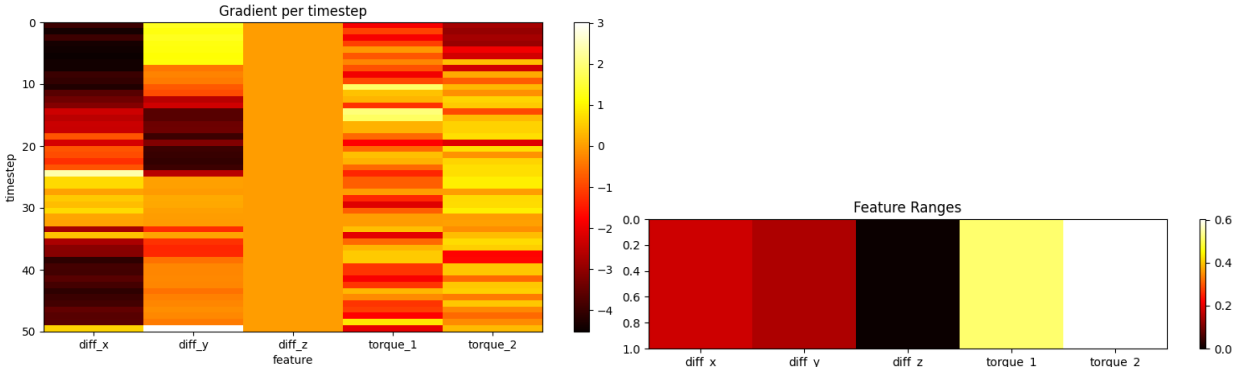
(a) Reacher, Success (b) Feeding, Success (c) Feeding, Reward (d) Itch Reward

Figure 3.8: **Distractor Features.** In general, removing distractor features that have no influence on preferences improves performance.

Table 3.2: **KL Divergence: Distractor Features.**

Env	With	Without
Feeding	4.732	**6.985**
Itch Scratching	**18.749**	8.570

features—namely, the difference between spoon position and target position. Thus, removing purely non-causal features fails to address this issue in Feeding.



(a) Gradients - W/out distractors (b) Feature Ranges - W/out distractors

Figure 3.9: **Saliency Maps: Without Distractor Features.**

Model Capacity

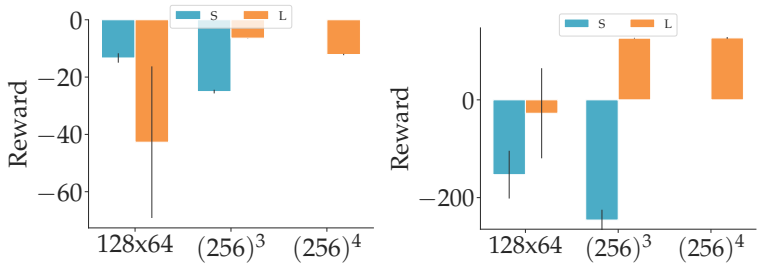
We study the effect of model capacity on the learned reward. We find that, despite careful tuning of hyperparameters with each model and dataset size, increasing the capacity of

Table 3.3: **EPIC Distances, Model Capacity.** The reward model with larger capacity (hidden layer dimensions of 256x256x256) appears closer to the ground truth than the reward model with smaller capacity (hidden layer dimensions of 128x64) *on the distribution of states seen during reward learning*, but is much further from the ground truth *on the distribution of states seen during reinforcement learning*.

RewL / RL	256x256x256	128x64
ground truth	0.210 / 0.707	0.233 / 0.598

the reward model does not necessarily result in an increase in subsequent policy performance.

We study the effect of model capacity on the learned reward. We find that, despite careful tuning of hyperparameters with each model and dataset size, increasing the capacity of the reward model does not necessarily result in an increase in subsequent policy performance. In fact, as seen in Fig. 3.10, only the Feeding task trained with the large (L) dataset size (52326 preferences) benefits from steadily increasing reward model capacity. Indeed, although increasing the capacity for the Reacher task appears to initially increase performance for the large dataset case (Fig. 3.10a), performance drops back down to below the performance of the smallest model size when the model size is further increased. Further, increasing model size seems to decrease the performance on the small datasets. This is not surprising, as benefiting from larger capacity tends to require increasing the amount of data. As such, the validation-set accuracies here tend to agree with learned reward performance.



(a) Reacher (b) Feeding

Figure 3.10: **Model Capacity.** Increasing model capacity increases performance when we use the large (L) dataset ($\binom{324}{2} = 52326$ preferences), to a certain extent. However, increasing capacity decreases performance on the small (S) dataset ($\binom{40}{2} = 780$ preferences).

Table 3.4: **KL Divergence Approximations, Model Capacity.** Increasing model capacity may result in a much larger distribution shift (as measured by the KL Divergence).

	128x64	256x256x256	128x64	256x256x256	256x256x256x256
Reacher	64.165	57.724	46.031	22.187	88.927
Feeding	25.352	32.836	16.900	4.732	9.909

To examine these results further, we compare the learned rewards directly with the ground truth reward (without performing policy optimization) using the EPIC pseudometric in Table 3.3. Interestingly, the EPIC distances between the learned rewards and the ground truth reward vary widely depending on the choice of coverage distribution. With the distribution of states seen during reward learning (generated by randomly switching between an expert and random policy), the larger (256x256x256) model appears closer to the ground truth reward than the smaller (128x64) one. However, these results are flipped when we instead evaluate the pseudometric on the distribution of states seen during policy optimization. This suggests that increasing the capacity of the reward model allows it to more closely 'mimic' the ground truth reward *on the reward learning distribution* (hence the lower EPIC distance) but worsens its ability to generalize to the reinforcement learning distribution. We see a similar effect with the divergence metric in Table 3.4.

Noise in Stated Preferences

Because user noise is often inevitable, we explore the effect of various types of user noise on reward misidentification. Following [101], we modify Eq. 3.1 to include a rationality constant β and a myopic discount factor $\gamma \in (0, 1]$:

$$P(\tau_A \prec \tau_B) = \frac{\exp(\beta \sum_{t=1}^{H} \gamma^{H-t} r(s_t^B, a_t^B))}{\exp(\beta \sum_{t=1}^{H} \gamma^{H-t} r(s_t^A, a_t^A)) + \exp(\beta \sum_{t=1}^{H} \gamma^{H-t} r(s_t^B, a_t^B))}. \tag{3.4}$$

Using Eq. 3.4, we explore four types of user noise (varied independently of each other): STOCHASTIC, where the user is rational with $\beta = 1$; MYOPIC, where earlier rewards are discounted with a $\gamma = 0.99$; SKIP, where the user skips a pair of trajectories if both have rewards below a certain threshold; and MISTAKES, where the preference label is randomly flipped with probability $\epsilon = 0.1$. ORACLE has $\beta = \infty$. Results are displayed in Fig. 3.12.

We find that noise in the stated preferences exacerbates reward misidentification—test accuracy stays high while policy performance plunges (Fig. 3.12a). Notably, two instances of user noise, Stochastic and Skip, have test accuracies greater than or equal to the Oracle (despite far worse performance). Importantly, the poor performance of the learned

Table 3.5: **Percentage (%) of Mislabeled Data Provided by a Stochastic User.** Numbers here correspond to Fig. 3.11b.

	Medium	Large
Stochastic	2.726	2.870

Table 3.6: **KL Divergence Approximations, Noise in Stated Preferences.**

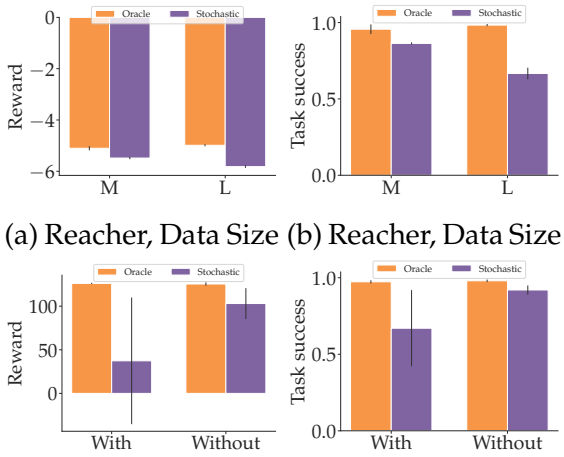
	Oracle	Stochastic
Reacher, Medium Dataset	6.762	**8.856**
Reacher, Large Dataset	7.365	6.739
Feeding, With Distractors	4.732	**13.188**
Feeding, Without Distractors	6.985	**8.046**

policies is not well predicted by the validation accuracies, with models achieving 0.995 resulting in worse alignment than models trained on less data with lower validation accuracy of 0.985.

Fig. 3.11 shows additional results. Increasing the amount of preference data when noise is present appears to have a negative effect (as seen in Fig. 3.11b), despite the proportion of mislabeled data remaining constant across dataset sizes (Table 3.5).

Further, there is a compounding effect with distractor features, where both noise and distractors result in a large loss in performance (Fig. 3.11d).

To examine these results further, we compare the learned rewards directly with the ground truth reward (without performing policy optimization) with the EPIC pseudometric in Table 3.7. Interestingly, the EPIC distances between the learned rewards and the ground truth reward vary depending on the choice of coverage distribution. With the distribution of observation-action pairs seen during reward learning (generated by randomly switching between an expert and random policy), the reward trained with labeling errors appears to be closer (in EPIC distance) to the ground truth than the reward trained without labeling errors. However, these results are flipped when we instead evaluate the pseudometric on the distribution of observation-action pairs seen during policy optimization. This suggests that the misidentified reward model more closely 'mimics' the ground truth reward *on the reward learning distribution* (hence the lower EPIC distance) but fails to generalize to the reinforcement learning distribution. Table 3.6 shows the KL divergences for each model configuration in Fig. 3.12.



(a) Reacher, Data Size (b) Reacher, Data Size

(c) Feeding, Distractors (d) Feeding, Distractors

Figure 3.11: **Noise in Stated Preferences.**

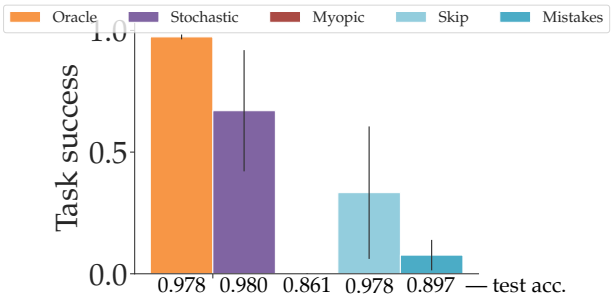
Table 3.7: **EPIC Distances, Noise in Stated Preferences.** Misidentified rewards (Stochastic) are closer to the ground truth reward on the reward learning distribution (RewL) but further from it on the RL distribution (RL) when compared to the properly learned rewards (Oracle). EPIC distances are computed on the rewards learned with distractors in the Feeding environment in Fig. 3.11d and averaged over three seeds.

RewL / RL	Ground truth	Stochastic	Oracle
Ground truth	0.000 / 0.000	**0.136 / 0.164**	**0.165 / 0.130**
Stochastic	**0.136 / 0.164**	0.000 / 0.000	0.220 / 0.209
Oracle	**0.165 / 0.130**	0.220 / 0.209	0.000 / 0.000

We use EPIC [54] as one way to measure the difference between reward functions. Table 3.7 shows that when the coverage distribution for EPIC is chosen as the distribution of state-action pairs seen during reward learning, the misidentified rewards due to user noise are closer to the ground truth rewards, in EPIC distance, than rewards learned without user noise. However, the opposite is the case when the coverage distribution is chosen to be the distribution of state-action pairs seen during RL. This implies that the misidentified reward model "mimics" the ground truth reward on the reward learning distribution but fails to generalize when taken out of distribution by RL.

We further observe in Table 3.6 that the KL divergence between reward learning and reinforcement learning state distributions is greater when the learned reward contains

noise (Stochastic). This indicates that the rewards trained with noisy data are misidentified: they have low test error but incentivize RL to deviate from the optimal behavior encountered during reward learning.



(a) Feeding, User Noise

Figure 3.12: **Noise in stated preferences** results in significantly degraded performance.

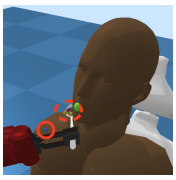
Partial Observability of Causal Features

Partial observability over aspects of the state on which the user's preferences are based is nearly inevitable in the real world. Interestingly, RL with the ground truth reward is able to learn proper feeding behavior *in spite of this lack of critical state information*. However, as displayed in Fig. 3.13, reward learning is only successful when the model is able to observe all the causal reward features.

We analyze gradient saliency plots and find that partial observability causes the reward model to incorrectly over-weigh causal features that are available and pick up on spurious correlations with non-causal variables. With Feeding, we observe that the reward model learns stronger weights on one of the joint angles and two components of the action, all of which have no bearing on the true reward. Simultaneously, the reward also learns a greater weight on the second component of the feature corresponding to the vector distance between the end effector and the mouth. This results in the behavior depicted in Fig. 3.13—the robot manipulator is able to successfully maneuver the spoon close to the patient's mouth (by observing the distance feature), but does so without ensuring that the food particles themselves stay on the spoon and end up in the patient's mouth.

Complex Causal Features

To explore how results may differ as we increase the complexity of the task, we evaluate reward misidentification on the complex task of Itch Scratching. We find that even after increasing the amount of training data for reward learning (Fig. 3.14a), increasing the reward model capacity (Fig. 3.14b), removing distractor features that are not causally



Observability	Performance Reward	Success
PREF-Full	126.120	0.973
PREF-Part	-120.825	0.040
GT-Part	128.933	0.990

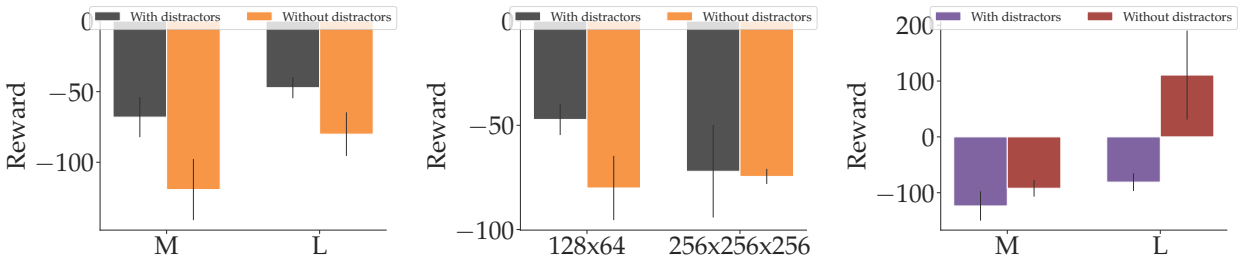
Figure 3.13: **Partial observability.** Minimizing spoon-mouth distance without observability of food leads to spilling food (left). PREF and GT refer to policies optimized with the learned and true reward, respectively. FULL and PART refer to the amount of observability over causal features.

related to the ground truth reward, having perfectly stated preferences, and ensuring full observability over all the features that are involved in the ground truth reward, performing preference-based reward learning still fails to produce a policy that successfully scratches the itch location on the patient. However, as shown in Table 3.1, a standard model-free RL algorithm trained using the ground truth reward is able to solve such a task.

The task's scratching motion requires not only making contact with the target using the end-effector, but also that the target contact position be greater than a δ away from the previous target contact position and that the exerted force be no more than a F_{max}. Although the preferences are based on how well this motion is performed and despite the reward model having access to all the necessary aforementioned information (including information about the state at the previous timestep; see Appx. 3.6), we find that the reward model is not able to learn the scratching motion. As seen in Fig. 3.14c, once we explicitly include a high-level indicator feature, "scratched" (whether the robot has successfully performed the scratching motion), performance drastically increases. We suspect that the reward model's tendency to pick up on spurious correlations that occur consistently over the course of the trajectory involving just a few variables prevents it from learning the true causal relation that involves many variables, each of which are causal only in a particular context.

We analyze this further and find that the learned reward without an explicit "scratched" feature leads to a greater amount of distribution shift. Future work should address the problem of learning this kind of complex, multi-feature causal relationship.

In the Itch Scratching task rewards the robot needs to perform a 'scratching' motion, which entails not only making contact with the target using the end-effector (being within a certain radius of the target coordinates), but also requires that the target contact position be greater than a certain δ away from the previous target contact position and that the exerted force be no more than a specified F_{max}. Although the preferences are based on how well the robot performs this scratching motion and despite the reward model having



(a) Increasing dataset size (b) Increasing model capacity (c) With 'scratched' indicator

Figure 3.14: **Complex causal features.** Learning the "scratched" feature from low-level causal features is difficult, despite increasing dataset size and model capacity. Performance significantly improves when given access to a high-level 'scratched' feature.

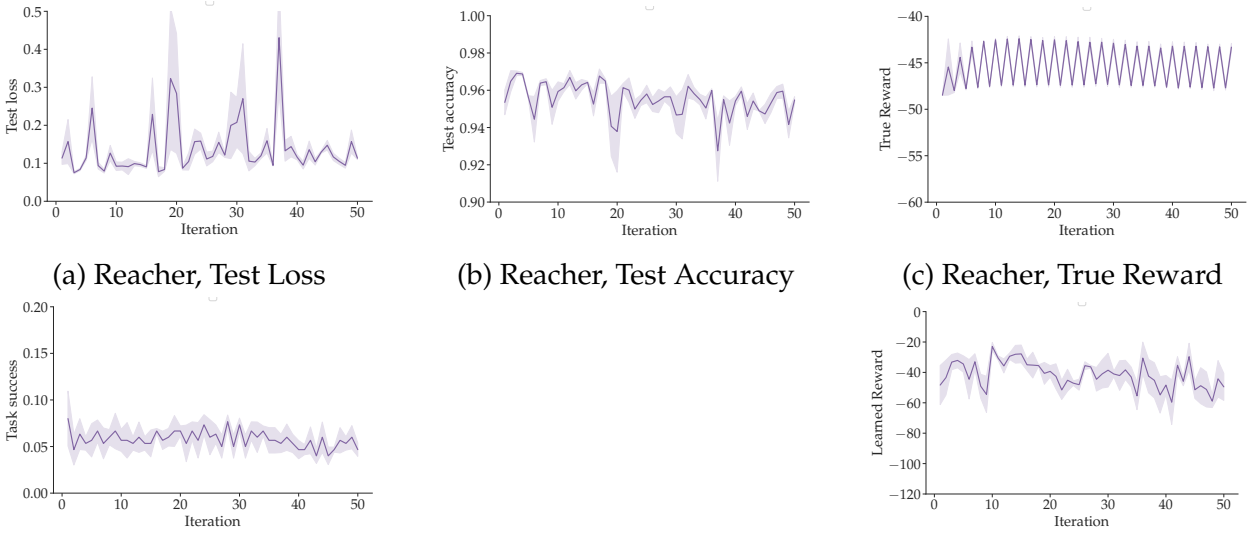
Table 3.8: **KL Divergence: Complex Causal Features.** Learning a reward without the 'scratched' indicator feature leads to a greater amount of distribution shift (KL divergence) during policy optimization.

With 'scratched'	Without 'scratched'
8.570	**15.553**

access to all the necessary low-level information to infer this scratching motion (including information about the state at the previous timestep; see Appx. 3.6 for details), we find that the reward model is not able to learn the scratching motion. As seen in Fig. 3.14c, once we explicitly include a high-level indicator feature, 'scratched', denoting whether the robot has successfully performed the aforementioned scratching motion, performance drastically increases. We suspect that the reward model's tendency to pick up on spurious correlations that occur consistently over the course of the trajectory involving just a few variables prevents it from learning the true causal relation that involves many variables, each of which are causal only in a particular context. This is supported by Fig. 3.5c, which shows gradients staying relatively constant for each feature rather than varying with time and context. As a result, the learned reward without an explicit 'scratched' feature leads to a greater amount of distribution shift. This is shown in Table 3.8 which shows the KL divergences with and without the hand-specified complex 'scratch' feature.

Iterative Preference-Based Reward Learning from Online Data

Finally, we recognize that active and iterative methods for data acquisition is also a widely popular solution—as such, we provide a preliminary exploration.



(a) Reacher, Test Loss (b) Reacher, Test Accuracy (c) Reacher, True Reward

(d) Reacher, Success (e) Reacher, Learned Reward

Figure 3.15: **Training the reward and policy together iteratively.**

Following [33], we verify the occurrence of reward misidentification when learning from data that is actively queried online (during the learning process). As before, we use the initial set of offline trajectory and preference data (acquired via Appx. 3.4) to train a reward function, which we then use to optimize a policy using RL. We then sample 10 trajectories from the most recently optimized policy and generate preferences for all possible pairs (1) between sampled and offline trajectories and (2) within the set of sampled trajectories, which we concatenate to our existing training dataset. Using the (now augmented) dataset, we fine-tune the reward function and policy for 10 epochs and 100 iterations, respectively. We repeat this process of taking rollouts from the most recent policy and fine-tuning the existing learned reward and policy for 50 iterations. Results are displayed in Fig. 3.15. We observe that reward misidentification appears to still be present—test error is very low, while resulting policy performance (measured by true reward and success) remains poor.

Our work cautions that reward learning is brittle—natural choices for acquiring data, deciding on the amount of data, or defining the input space can lead to models that seem very close to the true reward, but lead to spectacular failures when optimized. While active and iterative methods for data acquisition may alleviate some of these issues, learning reward models when some of the causal features are unobservable remains an open challenge.

3.7 Conclusion

Our work provides an analysis of reward misidentification and causal reward confusion. We identify three tasks where preference learning often results in misaligned learned reward models. Interestingly, these models have good validation and test accuracy—sometimes even 99.5%—and seem to distinguish the basics of task success versus failure. However, optimizing these rewards via RL pushes the policy outside of the training distribution, where the model falsely believes the reward is higher—we demonstrate this via saliency maps, EPIC distance, and KL divergence and by examining the resulting behaviors. This out-of-distribution effect results in policies that achieve high learned rewards but have poor true rewards. We find that it is easy for reward models to pick up on non-causal features, as some issues go away when we eliminate these non-causal features from the input. Furthermore, noisy preference data aggravates poor generalization. And when not all causal features are observable, the learned reward model will struggle even though there exists a high-performing policy that only uses observable state information.

Based on our results, we have identified several directions for future work. First, our results show that reward misidentification induces a distribution shift such that the learned reward appears deceptively close to the true reward on the training distribution, despite leading to misaligned behavior when optimized via RL. As such, future work should investigate methods for penalizing excess exploration beyond the training data distribution. We examine the effect of adding a direct penalty on the KL divergence into the cost function during RL. We find that the RL agent is again able to hack the learned reward—performance becomes even worse because of the additional degree of freedom in the reward function afforded by the discriminator penalty. However, we hypothesize there are other ways to successfully incorporate such a penalty. Next, our results demonstrate that spurious ("nuisance") features can significantly increase the chance of reward misidentification. Thus, in cases where we can query for human knowledge on exactly which features are spurious or causal, we can use this feedback to learn which features are non-causal or remove non-causal features entirely. Further, our results indicate that high-level features such as the "scratch" feature are difficult for neural networks to learn, even when all the necessary low-level information is available. Future work should examine incorporating methods for learning high-level features [14]. Using alternatives to EPIC [54] such as DARD [202] to compare learned rewards may also prove fruitful in detecting and alleviating reward misidentification.

Chapter 4

Active Learning of Human Preferences

Given that the reward function is often learned on a finite set of environment, this introduces an inherent limitation that the reward function may not generalize to new scenarios.

Why Active Learning for Human Preferences?

One popular way to design robotic agents is to program them to maximize the cumulative reward in their environments.

While we have made great progress in solving the reward optimization problem, we often delegate finding a good reward function to the human designer. In reality, reward design is an iterative and challenging process. Human designers spend tremendous effort iterating on the reward function as they test the robot in more and more environments. What is more, once deployed, reward functions can still be incorrect in edge-case environments, leading to failures.

Take, for instance, autonomous cars. They have to correctly balance safety, efficiency, comfort, and abiding by the law. A human designer starts by looking at some representative traffic scenarios and specifying a reward function that leads to desired behaviors in each of them. But working well in this set of environments is not enough — the reward function has to incentivize the right behavior in *any* environment the car will encounter in its lifetime[1]. The engineer will test-drive the car in both simulation and in the real world. Almost inevitably, optimizing for the specified reward will lead to some undesirable behavior in some further environments. The engineer will then *revise* the initial reward and repeat the process. If they are lucky, they will have converged to a suitable reward function by deployment time. But for many systems, this is a never ending process: the car will encounter some edge-cases in the real world that they did not foresee — an unseen road layout or traffic situation — and the engineer revises the reward again on the new environment.

[1]While there are challenges to building a world model, planning algorithms, etc., for this paper, we assume success in that and focus on designing the reward function.

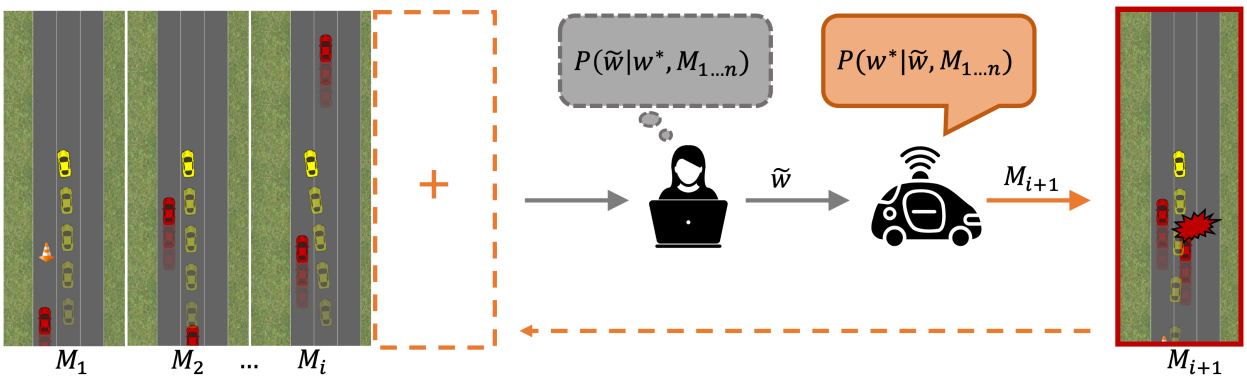


Figure 4.1: In the Assisted Reward Design process, the designer specifies a proxy reward $\tilde{w}$ on a set of environments $M_{1,\dots n}$. The robot agent takes the current proxy, infers about the true reward w^* and queries the designer with a new environment where the proxy fails. The designer then revises her proxy to work on this new environment.

In this work, we explore what it would mean for the robot to explicitly account for this iterative nature of reward design. Rather than treating the current reward as set in stone, we envision a system in which the robot works together with the designer and helps them design the right reward function, as in Fig. 4.2: the robot takes the reward function specified by the designer, and proposes a new environment for the designer to investigate where the reward function might fail. To capture this formally, we first enable the robot to explicitly account for the *uncertainty* in the reward function itself: every reward revision does *not* define the reward function fully, but is instead an *evidence* about it. Previous work on Inverse Reward Design [64] incorporates this uncertainty into the reward function with the goal of making the robot plan more conservatively in new environments. In this work, rather than passively updating the robot's estimate of the reward, we argue that it is the robot's job to actively help the engineer design more robust reward functions, with the goal of performing well at deployment time. We refer to this as *Assisted Reward Design*. The key insight is that:

By accounting for the iterative nature of reward design, the robot does not just passively estimate the reward it receives but proactively influences the future evidence it can gather from the designer.

Our contributions are three-fold:
1. Formalizing reward design as an iterative process. We formalize the problem of iterative reward design as a Meta MDP with hidden state. Under this formulation, the robot can and should account for the future iterations of the reward as *future evidence*. The robot can then make these pieces of evidence more informative by exposing the designer to environments where revisions are most likely needed. Rather than letting the designer eventually encounter edge-case environments, our method *proposes* candidate environ-

ments and asks for reward designs, inserting itself in the design process loop, as in Fig. 4.2. This hps two implications. In cases where the designer would converge to a good reward before deployment time, our method speeds up the iterations by focusing on informative environments. Perhaps more importantly, in cases where the designer might still otherwise have the wrong reward when deploying the system, our method has the potential to expose the edge-case environments that cause failures, prompting fixes in the reward ahead of time.

2. Providing an approximate algorithm for our method. It is intractable to compute the reward uncertainty given the high-dimensional nature of the reward space and the environment space. We propose an approximate algorithm based on particle filtering to enable efficient computation.

3. Analyzing Assisted Reward Design in a simplified autonomous car domain. We evaluate our method's practicality and effectiveness in designing reward functions for autonomous cars, using the open-source environment proposed by [155]. We conduct experiments with both simulated reward designers and robotics experts. We find that our method uncovers more difficult test scenarios and speeds up the reward design process. We also showcase that it proposes interesting environments that tend to be *edge cases* where the current reward estimate fails.

Overall, this paper takes a step towards making our robotic systems understand reward specification as an iterative process. In the end, we would like robotic systems not just to optimize what we design for them but help us design better and more robust reward functions. Even though admittedly more work is needed to put these ideas in real-world systems, especially in light of the need for a simulator or learned model of environments, we are excited to see our method assist in difficult reward design tasks and produce *edge-case environments* without any hand-coded heuristics.

4.1 Problem Statement

Active Reward Learning Our work is an instance of active reward learning. Prior work has focused on actively querying for comparisons [12, 188], demonstrations [23, 42, 41, 110], or reward labels [148]. Such queries are typically for trajectories or states in a given environment (MDP). Recent work has proposed active learning to synthesizes the environment itself to gain more information about the reward (either the initial configuration and behavior of other agents in the scene, such as [154], or the entire environment [148]). We also propose to actively query new environments, but the human feedback is a proxy reward designed to work only in that environment. This is interesting for two reasons. First, we show that it leads to the robot proposing edge cases to the designer where the previous reward is likely to fail, thus aiding in revising the reward. Second, we argue that the same active reward learning ideas from the literature can be applied to the reward engineering process — when we design a reward function, we should not just have

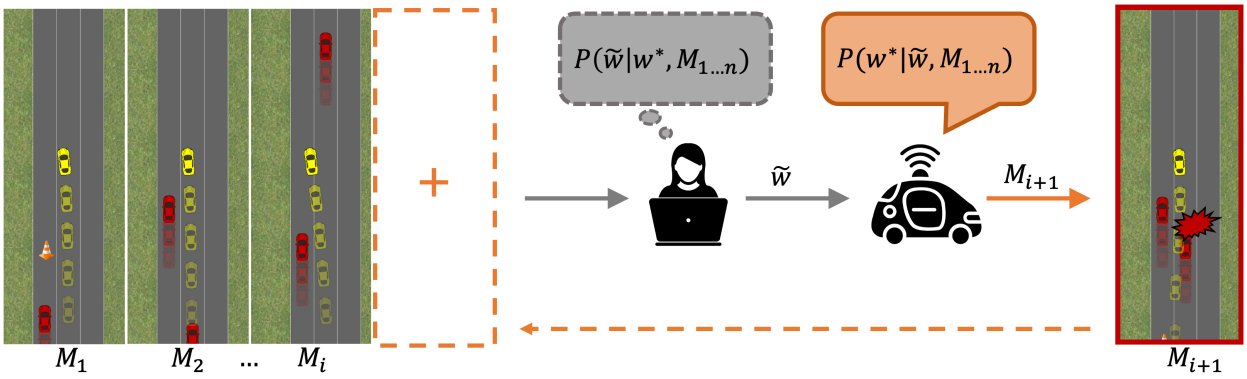


Figure 4.2: In the Assisted Reward Design process, the designer specifies a proxy reward $\tilde{w}$ on a set of environments $M_{1,\ldots n}$. The robot agent takes the current proxy, infers about the true reward w^* and queries the designer with a new environment where the proxy fails. The designer then revises her proxy to work on this new environment.

the robot go optimize it blindly. We should instead think about it as an iterative process through the lens of active reward learning.

Reward Shaping Many prior works have also focused on recovering reward functions for suboptimal planners, given a ground truth reward function that is difficult to optimize, e.g. due to sparsity or non-differentiability [167, 149]. Note that in many real-world problems, such as autonomous cars, the ground truth reward is not obvious to write down. Assisted Reward Design is different from the reward shaping problem in that we do not assume access to a ground-truth reward function, but seek techniques that leverage proxy rewards that are the designer's best guesses.

Problem Setup Let an "environment" M be a Markov Decision Process without the reward function. At development time, we assume access to a large set of such environments, $\mathcal{M}_{\text{devel}}$. We assume this is a large set that the designer cannot exhaustively test. This can be, for instance, all the environments in a several-million-mile autonomous driving dataset. Or, it can be a parameteric set of possible environment configurations, such as placements of objects, roads, and other agents. Finally, it can also be a generative model of the world. The robot will run at deployment time in a different set of environments $\mathcal{M}_{\text{deploy}}$, which we do *not* have access to during development time. [2]

We further assume access to a space of reward functions parameterized by $w \in W$ — these can be linear weights on pre-defined features or weights in a neural network mapping raw input to scalar reward. We denote by w^* weights of the ground truth reward function, which induces the desired behavior in $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$. We denote by $\xi_{w,M}$ the trajectory (or policy, but for this work, we consider deterministic MDPs with set initial

[2]Our method works best when we can induce a vast set $\mathcal{M}_{\text{devel}}$ that includes all possible scenarios in $\mathcal{M}_{\text{deploy}}$. Otherwise, if $\mathcal{M}_{\text{deploy}}$ is allowed to be drastically different, and if $\mathcal{M}_{\text{devel}}$ is not enough to design a robust reward function, this exposes the robot to unanticipated failures after deployment.

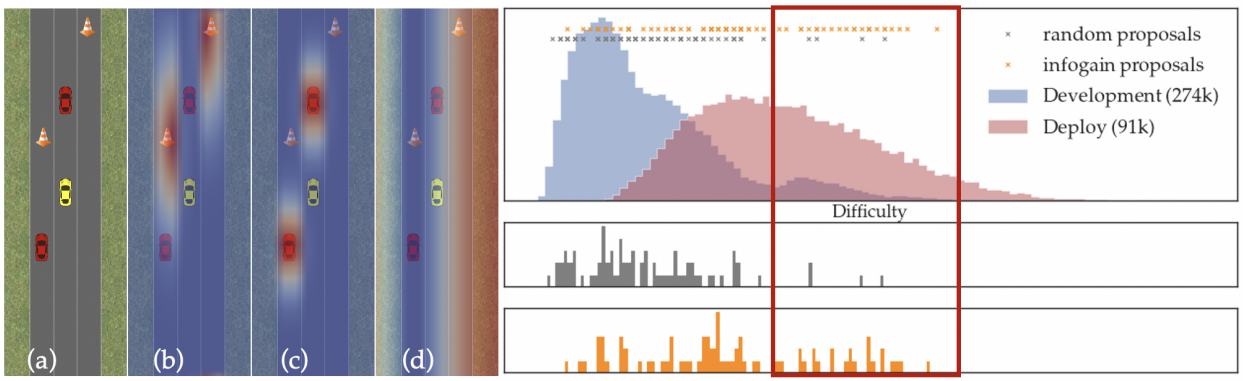


Figure 4.3: Left: (a) The driving environment with autonomous vehicle (yellow) and human-driven vehicles (red). (b)(c)(d) highlights selected features: obstacle distance, vehicle distance, progress. Right: the distribution of $\mathcal{M}_{\text{devel}}$ (blue) and $\mathcal{M}_{\text{deploy}}$ (pink) based on difficulty metric.

states) optimal to the cumulative reward induced by w in M, $\xi_{w,M} = \arg\max_{\xi} R_w(\xi; M)$. Our assumption is that there exists a w^* such that $\xi_{w^*,M}$ is the desired behavior for any $M \in \mathcal{M}_{\text{devel}}$ and any $M \in \mathcal{M}_{\text{deploy}}$. We do *not* have access to w^*.

The Process of Unassisted Reward Design The designer takes a subset of environments $\mathcal{M} \subseteq \mathcal{M}_{\text{devel}}$ and specify a reward function (via parameters $\tilde{w}$) that leads to good behavior in those environments. In a one-shot design, they would select a set $\mathcal{M}$, assume it is representative of $\mathcal{M}_{\text{deploy}}$, design a $\tilde{w}$ for that $\mathcal{M}$, and deploy the system with $\tilde{w}$ (never changing the reward again). However, in reality, this is an iterative process. They start with an $\mathcal{M}_0$, design a $\tilde{w}_0$, and eventually encounter during their testing a new environment M' on which optimizing $\tilde{w}_0$ does not lead to desirable behavior. Then, they would augment their set to $\mathcal{M}_1 = \mathcal{M}_0 \cup \{M'\}$, and re-design the reward: $\mathcal{M}_0 \rightarrow \tilde{w}_0 \rightarrow \mathcal{M}_1 \rightarrow \tilde{w}_1 \rightarrow \ldots$. This might continue into deployment if the reward at the time of deployment still fails on $\mathcal{M}_{\text{deploy}}$. Implicitly, at every step along the way, the robot treats the current $\tilde{w}_i$ as equivalent to w^*.

The Assisted Reward Design Problem

We formulate the Assisted Reward Design Problem by introducing a meta-MDP problem, where the meta-agent receives proxy rewards as observations from the designer and acts by proposing new environments. The goal is to minimize regret of the reward design over deployment $\mathcal{M}_{\text{deploy}}$.

Meta-States, Meta-Actions, Observations, Transitions.[3] In the Assisted Reward Design problem, the notion of a "state" is a set $\mathcal{M}$ of environments — this is what changes over

[3]For the rest of this paper, all our references to "state" and "action" stand for "meta state" and "meta action".

time and is directly observable — along with the hidden state w^*. The meta-agent starts at state $(\mathcal{M}_0, w^*)$ (environments either chosen by the designer to be representative, or simply the empty set). It transitions to the next state by "acting", i.e. adding one more environment M_i to the set: $\mathcal{M}_{i+1} = \mathcal{M}_i \cup M_i$. Every time it enters a state $(\mathcal{M}_{i+1}, w^*)$, it receives an observation $\tilde{w}_{i+1}$ about w^* from the designer. For the meta-agent to assist the designer, it needs to treat the observed reward $\tilde{w}_i$ not the same as w^*, but as a proxy that emulates w^* *only* on the current $\mathcal{M}_i$. It can then interpret $\tilde{w}$ as evidence about w^* and use all past evidence to obtain a belief of w^*. To assist the designer, the key step is to account for (and thus influence) the future evidence in order to get a better reward estimate.

Observation Model. Reward designers tend to select proxy rewards that work well in training environments. With this intuition, previous work [64] introduced an observation model that computes the probability of observing proxy reward $\tilde{w}$ under w^* and designer's training environment set $\mathcal{M}$:

$$P_{\text{design}}(\tilde{w}|w^*, \mathcal{M}) \propto \exp\left[\sum_{i=1}^{|\mathcal{M}|} \beta R_{w^*}(\xi_{\tilde{w}_i, M_i})\right] \tag{4.1}$$

with β controlling how close to optimal we assume the person to be. This distribution informs the meta-agent that it is receiving proxy rewards from an approximately optimal designer who looks at the current $\mathcal{M}$. We can then leverage this model to infer a belief over w^*: $b_{\mathcal{M}}(w^*)$.

Objective. The objective of Assisted Reward Design is to achieve high reward at deployment time. What the meta-agent controls are a sequence of actions, i.e. a sequence of environments $\mathcal{M}_T$ it can propose. These lead to observations about w^*. At deployment time, the meta-agent uses its belief to generate optimal trajectories and accumulates reward according to w^*. We want to find the set of environments $\mathcal{M}_T$ such that when given to the designer, these environments induce a belief that leads to the best performance in $\mathcal{M}_{\text{deploy}}$:

$$\max_{\mathcal{M}_T} \mathbb{E}_{M \sim \mathcal{M}_{\text{deploy}}} \mathbb{E}_{w \sim b_{\mathcal{M}_T}(w^*)} R_{w^*}(\xi_{w,M}) \tag{4.2}$$

where $b_{\mathcal{M}_T}(w^*)$ is the meta-agent's belief at deployment time, based on the evidence gathered from the states $\mathcal{M}_0, .., \mathcal{M}_T$; $\xi_{w,M} = \arg\max_\xi R_w(\xi; M)$ is the optimal trajectory.

As currently stated, this objective is impossible to optimize directly because the meta-agent does not know $\mathcal{M}_{\text{deploy}}$. In what follows, we introduce a solution to Assisted Reward Design based on the heuristic objective of identifying w^* — if the meta-agent uses its actions to disambiguate what w^* should be, then it can achieve optimum performance on the objective in Eq. (4.2) because it can plan optimally to w^*. We first introduce the Maximal Information objective. Then we show how the meta-agent can track a belief and perform future belief updates in order to propose new environments.

4.2 Assisted Reward Design via Active Info-Gathering.

Maximal Information. We invert the probability model of Eq. (4.1) to obtain a belief over w^*: $P(w^*|\tilde{w}, \mathcal{M})$. Our key insight is that we can reduce the belief uncertainty by proposing future environments to influence the designer. In other words, we would like to find environments that can disambiguate the reward as much as possible — if we can identify w^* exactly, the meta-agent attains zero regret at deployment. Thus, we introduce an approximate strategy for the meta-agent: proposing environments that provide the most information gain over its belief distribution, as commonly done in active learning [73, 51] and robot active exploration [26]. For computational tractability, we do a one-step look-ahead only, although the meta-agent could benefit from reasoning over multiple future iterations. Given an action M (MDP), we compute its mutual information with w:

$$f(M_{i+1}) = \mathbb{I}[w; M_{i+1}] = \mathbb{H}[w|\mathcal{M}_{0:i}, \tilde{w}_{0:i}] - \mathbb{E}_{w^* \sim P_i(w), \tilde{w}_{i+1} \sim P_{\text{design}}(\tilde{w}_{i+1}|w^*, \mathcal{M}_{i+1})}[\mathbb{H}[w|\mathcal{M}_{0:i+1}, \tilde{w}_{0:i+1}]] \tag{4.3}$$

Here $\mathbb{H}[w|\mathcal{M}_{0:i}, \tilde{w}_{0:i}]$ is the entropy of posterior distribution of $P_i(w)$, and $\tilde{w}_{i+1}$ is a new observation sampled from the updated designer model $P_{\text{design}}(\tilde{w}_{i+1}|w^*, \mathcal{M}_{i+1})$. After observing $\tilde{w}_{i+1}$, we can compute the conditional entropy $\mathbb{H}[w|\mathcal{M}_{0:i+1}, \tilde{w}_{0:i+1}]$ using methods from Sec. 4.2. In our implementation, we set $\tilde{w}_{i+1} = w^*$ to make Eq. (4.3) more tractable and find that it empirically works well. Next we introduce two update rules to compute conditional entropy — joint and independent mode.

Updating Beliefs over w^*

Joint reward design. At iteration $i + 1$ of Assisted Reward Design, the meta-agent takes an action M_{i+1} (proposes an environment), and receives an observation $\tilde{w}_{i+1}$ – a reward that works over *all* environments $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{M_{i+1}\}$. Assuming that $\tilde{w}_{i+1}$ is independent from previous observations, we can use $\tilde{w}_{i+1}$ to update the belief over w using Eq. (4.1):

$$P_{i+1}(w|\tilde{w}_i, \mathcal{M}_i) \propto P_i(w) P_{\text{design}}(\tilde{w}_i|w, \mathcal{M}_i) \tag{4.4}$$

with P_0 the meta-agent's prior (uniform in our experiments). We take a sequential importance resampling particle filtering approach to approximate P_{i+1}: we sample N_p particles from P_i, compute importance weights based on Eq. (4.1) for each particle, and resample using the importance weights. In order to compute the normalized P_{design}, we approximate the normalizer in Eq. (4.1) via samples.

Independent reward design. We also allow designers to provide an observation $\tilde{w}_i$ based solely on the new environment M_i. Prior work [147] finds that this model puts less burden on the designer. Under this assumption, our belief update simplifies to relying on a single new environment:

$$P_{i+1}(w|\tilde{w}_i, M_i) \propto P_i(w) P_{\text{design}}(\tilde{w}_i|w, M_i) \tag{4.5}$$

Adding New Features. When the designer adds in new feature ϕ_{k+1} on environment M_n, the new proxy reward has $d+1$ dimensions while the previous proxies have d dimensions. We can still perform reward inference on all of the $d+1$ dimensions, as long as we incorporate all proxies received so far. The meta-agent would then have to revisit all the reward designs it's gotten so far and recompute posterior over the new augmented space. To do so, we invert Eq. (4.1) using the formula from [64]:

$$P(w = w^*|\tilde{w}_{1:n}, \tilde{M}_{1:n}) \propto P(w) \prod_{i=1:n} \frac{\exp\left(\beta w^T \tilde{\phi}_i\right)}{\tilde{Z}}, \tilde{Z}(w) = \int_{\hat{w}} \exp\left(\beta w^T \hat{\phi}_i\right) d\hat{w}$$

with $P(w)$ being the prior. We use MCMC to infer the distribution. During inference, every sample w is of $d+1$ dimensions, and $\tilde{\phi}_i$ are of $d+1$ dimensions, computed from all previous proxy rewards (of i or $i+1$ dimensions) on the existing tasks. To compute the normalizer, we integrate over $\hat{w}$ in the $d+1$ dimensional space.

Algorithm

Algorithm. We now present our main algorithm for Assisted Reward Design. At every iteration i, we use the Maximal Information acquisition functions to select the next environment M_{i+1} and query the reward designer. Note that the environment space $\mathcal{M}_{\text{devel}}$ is continuous and high dimensional and it is intractable to exhaustively search through it. We thus use *uniform sampling* to select a candidate set $\mathcal{M}_{\text{cand}} \in \mathcal{M}_{\text{devel}}$. Note that one can employ different heuristics for selecting such candidate sets, and we leave the heuristic design to future work.

Representing Belief Distribution. We use particles to represent $\tilde{P}_i(w = w^*)$: at each step, we sample N_p particles from $\tilde{P}_i(w = w^*)$, compute importance weights based on Eq. (4.1) for each particle, and resample using the importance weights.

Algorithm 2 Assisted Reward Design via Info-Gathering

Require prior $P_0(w)$, $\mathcal{M}_{\text{devel}}$, N_{cand}, N_p, initial training environments $\mathcal{M}_0$
Initialize posterior $\tilde{P}_0(w = w^*) = P_0(w)$
for $i = 0, .., T$ **do**
  $\tilde{w}_i \sim P_{\text{user}}(w|w^*, \mathcal{M}_i)$ { Query the designer on $\mathcal{M}_i$ }
  Compute posterior $\tilde{P}_{i+1}(w = w^*)$ using Eq. (4.4) or Eq. (4.5)
  Sample N_{cand} candidate environments $\mathcal{M}_{\text{cand}} \subseteq \mathcal{M}_{\text{devel}}$
  for $M \in \mathcal{M}_{\text{cand}}$ **do**
    Compute $f(M)$
  end for
  Select $M_{i+1} = \arg\max_{M \in \mathcal{M}_{\text{cand}}} f(M)$
  $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{M_{i+1}\}$
end for

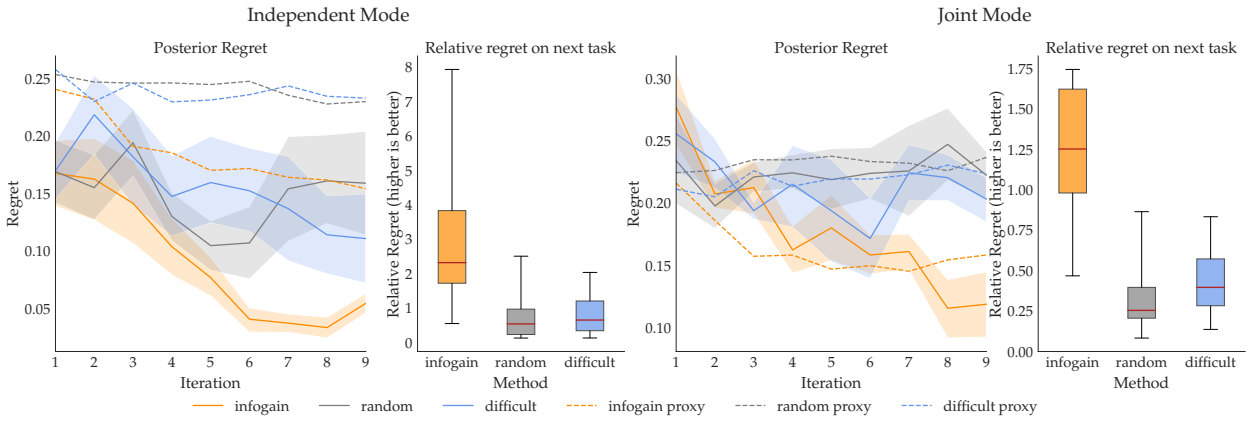


Figure 4.4: Experiment with simulated designers. Left: independent mode, right: joint mode. (a)(c) shows the regret of the posterior mean compared to true reward under different acquisition functions. (b)(d) shows the efficacy of the proposed environment. The "box" denotes 5th to 95th percentile.

Complexity. Our algorithm is bounded by the number of particles N_p for representing posterior distribution and the number of candidates N_{cand}. At every iteration, the algorithm needs to solve for each particle w in each candidate environment, which leads to a total of $\mathcal{O}(N_p \cdot N_{\text{cand}})$ planning problems in order to compute belief update. This is the main speed limit. While one can potentially speed up by learning fast planners [159], we leave this to future work. In our experiments, we implement the environment as in Sec. 4.3 such that the dynamics and reward function are both vectorizable. We then concatenate the reward functions of $\mathcal{O}(N_p \cdot N_{\text{cand}})$ problems and compute batch forward planning using gradient-based planner. Note that this is not feasible for general planning problems with non-differentiable dynamics or reward functions.

4.3 Experiments with Simulated Designers

We evaluate Assisted Reward Design in a simplified autonomous driving task. In this section, we hypothesize a ground truth reward function and simulate designer behaviors. Note that our method does not require such unique ground truth w^* – this is merely used here for evaluation. In Sec. 4.4, we conduct an expert study where we do not have the ground truth.

Experiment Setup

This section provides the details of the driving environment used in the paper. We introduce definitions of environment dynamics, feature and reward functions, the envi-

ronment distributions, and how we implement the environment efficiently for trajectory optimization.

Dynamics We represent the forward dynamics of the MDP as $x_{t+1} = f(x_t, u_t)$, where we have the full knowledge of x. To model system dynamics, we use simple point-mass vehicle model with holonomic constraint. Let $x_{\text{car}} = [x, y, \theta, v]^\top$, where x, y are the corrdinates of the vehicle, θ is the heading, and v is the speed. We let $u = [u_{\text{steer}}, u_{\text{acc}}]^\top$ be the control input, where u_{steer} is the steering input and u_{acc} is the acceleration. We also use α as the friction coefficient. The model for a single vehicle is:

$$[\dot{x} \;\; \dot{y} \;\; \dot{\theta} \;\; \dot{v}] = [v \cdot \cos(\theta) \quad v \cdot \sin(\theta) \quad u_{\text{steer}} \quad u_{\text{acc}} - \alpha \cdot v] \tag{4.6}$$

We model human vehicles as moving forward at constant speed and traffic cones as static objects.

Driving task. As illustrated in Fig. 4.3, the autonomous car is to merge to the leftmost lane. Since there may be other human vehicles (simplified as constant speed) and obstacles, the autonomous car needs to decide whether to overtake or to slow down. Our reward function based on [155] is a weighted sum of six features, including vehicle distance, progress, control effort, etc. A good reward function should correctly balance these terms and reflect our desired behaviors in all different situations. To obtain $\mathcal{M}_{\text{devel}}$, we parametrize environments over the initial placements of obstacles and other vehicles.

Feature and Reward Functions. In the following section 4.1 we introduce the environment features ϕ in the driving environment. Each feature composes of ϕ_{raw}, a subset of the current state x_t and control u_t, and a non-linear transformation on ϕ_{raw}. The nonlinear transformation is designed such that when maximized, it induces the desired behaviour in that feature, such as moving to the target lane. These environment features are are differentiable and that we can characterize the desired driving behavior via a linear weighed sum $w^\top \phi$.

Environment Difficulty. Successful merging tends to be more difficult when human vehicles and obstacles are closer to the autonomous vehicle. We can use a simple metric to describe each environment: difficulty $= \sum_i 1/d_{\text{human vehicle }_i} + \sum_j 1/d_{\text{traffic cone}_j}$. We visualize the distribution of $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$ in Fig. 4.3 using this metric. Note that both $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$ have a "long-tail" distribution of events — events that are rare to encounter daily, but could be highly useful for the reward design. An ideal acquisition function in should be able to identify these events.

Baselines. We compare our Maximal Information acquisition function with two baselines that represent common design paradigms where the designer selects (1) the most difficult tasks ranked by $M \in \mathcal{M}_{\text{devel}}$ (2) random environments from $\mathcal{M}_{\text{devel}}$.

Experiment Results

Evaluation. A good Assisted Reward Design method should help the reward designer quickly recover high-quality reward functions. To evaluate this, we specify one set of

Environment Features			
Feature Name	Raw Feature ϕ_{raw}	Transformation ϕ_{full}	Meaning
Speed	v	$-(\phi_{\text{raw}} - v_{\text{goal}})^2$	How much the vehicle deviates from the goal speed.
Control	$[u_{\text{steer}}, u_{\text{acc}}]$	$-\|\|\phi_{\text{raw}}\|\|^2$	The control effort by the vehicle.
Lane	x	$\mathcal{N}(\phi_{\text{raw}} - \vec{x}_{\text{goal}}, d_{\text{lane}})$	How much the vehicle deviates from target lane center.
Car	$[x, y]$	$-\sum_i \mathcal{N}(\phi_{\text{raw}} - \vec{x}_{\text{car i}}, d_{\text{car i}})$	How much the vehicle is driving close to other vehicles.
Obstacle	$[x, y]$	$-\sum_i \mathcal{N}(\phi_{\text{raw}} - \vec{x}_{\text{obs i}}, d_{\text{obs i}})$	How much the vehicle is driving close to the obstacles.
Fence	x	$-[x_{\text{fence left}} - \phi_{\text{raw}}]_+ - [\phi_{\text{raw}} - x_{\text{fence right}}]_+$	How much the vechile is outside the fence

Table 4.1: Environment Feature Table

ground truth reward w^*. At every iteration, we simulate the designer using w^* under both *independent* mode and *joint* mode as described in Sec. 4.2.

Experiment Details

Optimal Control For the autonomous driving task, we compute trajectories of 10 timesteps. We use finite-horizon optimal planning with regard to given rewards. We plan at a horizon of 10 timesteps, and replan every 5 timesteps.

Planning Speed. We implement the dynamics and reward function using JAX [17] and leverage the JIT compilation to speed up running time. We use shooting method and perform gradient-based planning using the Adam optimizer for 200 steps. We plan at a horizon of 10 timesteps and replan every 5 timesteps. When planning for a single environment, we can generate 2.57 trajectories per second. Because we can vectorize the planning problem and plan for multiple trajectories at once, we can achieve 157.72 trajectories per second, by planning for 500 trajectories in batch. Computations are benchmarked on the c2-standard-4 instance on Google Cloud. We use batch planning for computing the belief

distribution and environment proposals, as discussed in Sec. 5.3 as the main bottleneck of our algorithm.

Reward Design We use rationality factor $\beta = \frac{0.1}{||\mathcal{M}||}$ to simulate noisy designer, and $\beta = 1$ in the inverse model. This is because we find empirically that the proxy reward quickly approaches the ground truth reward when we have multiple proxy environments. We thus divide the rationality factor β it by the number of proxy environments to maintain its noisiness. To compute the posterior probability in Eq. (4.4) or Eq. (4.5), we need to approximate the normalizing constant. We follow the approach in [64] by sampling w. In the section 4.2 of [64], they find empirically that it helped to include the candidate sample w in the normalizing sum. This requires planning with w and computing its feature sum in the MCMC inner loop, which largely slows down the inference. Instead, we include the candidate w, but multiplies it with proxy $\tilde{w}$'s feature sum in the normalizing constant.

We then compare three methods (random, difficulty, maximal information) in iterative fashions. We use the same initial environment for the three methods and perform 9 iterations. We aggregate results over five random seeds.

Environment Distribution. We use a simple method to define the distribution of environments $\mathcal{M}_{\text{design}}, \mathcal{M}_{\text{deploy}}$. There are two human vehicles and two traffic cones positioned on the three-lane highway. We assume that the autonomous vehicle starts from the center of the scene $(x = 0, y = 0)$, and sample the starting position of the other vehicles and obstacles. The other vehicles can start anywhere in $x_{\text{min car}} \leq x_{\text{car}} \leq x_{\text{max car}}, y_{\text{min car}} \leq y \leq y_{\text{max car}}$ and the obstacles can be initialized anywhere in can start anywhere in $x_{\text{min obs}} \leq x_{\text{obs}} \leq x_{\text{max obs}}, y_{\text{min obs}} \leq y \leq y_{\text{max obs}}$. We filter out situations where the other vehicles or obstacles are initialized to be colliding with the main vehicle.

Note that this is a very coarsely designed distribution to showcase our method. We have several limitations. For instance, we do not exclude the environments where the other vehicles runs into obstacles. These are relatively unlikely environments, and in principle we can define more realistic distributions by more careful environment engineering or by loading real world driving datasets.

The difference between $\mathcal{M}_{\text{devel}}$ and $\mathcal{M}_{\text{deploy}}$ is that in $\mathcal{M}_{\text{deploy}}$, we define a tighter feasible range of x_{car} centered around the autonomous vehicle. This results in a shifted distribution of pseudo-difficulty metric with long-tail events. After discretization, $\mathcal{M}_{\text{devel}}$ has 274k environments and $\mathcal{M}_{\text{test}}$ has 91k environments. Given these large number of possible environments, it is impossible for reward designers to enumerate them manually.

Environment Proposal. During environment proposal, we use $N_p = 500$ particles on $N_{\text{cand}} = 64$ environments. We implement a vectorized car dynamic model using JAX [17] and Ray [125] for batch planning. Our environment proposal step takes 4 minutes on c2-standard-4 instance on Google Cloud.

Evaluation. To examine the quality of our inferred posterior of the reward function, we need to evaluate the posterior in new driving environments. We thus sample 500 environments uniformly from $\mathcal{M}_{\text{deploy}}$ as the deployment set for evaluation. We compute the regret, note that because of the different placement of vehicles and objects, different en-

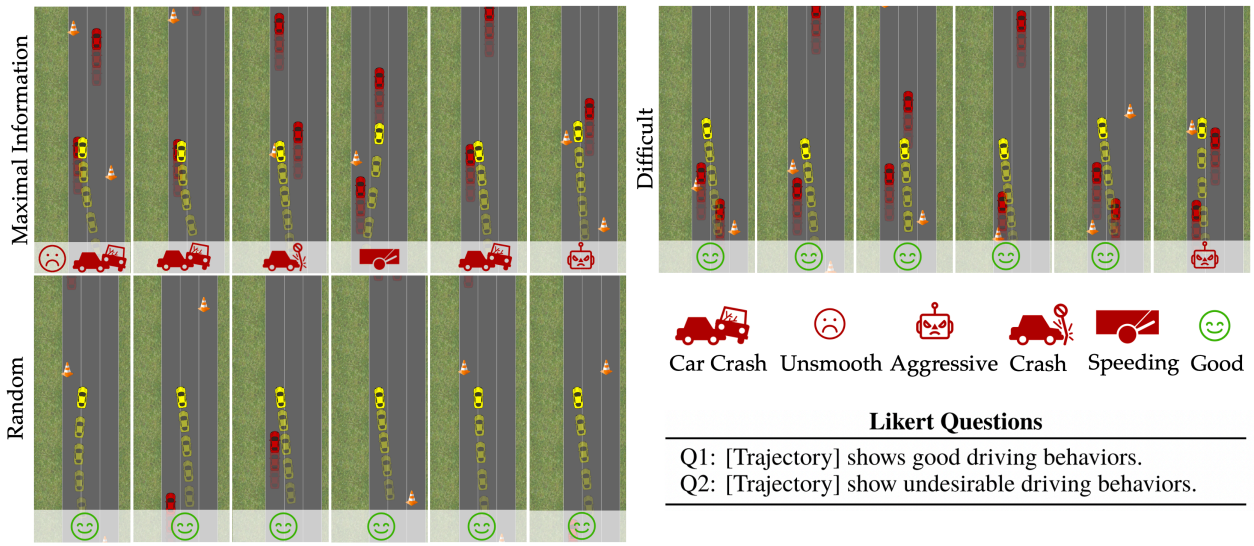


Figure 4.5: Visualization of the top proposed environments based on each acquisition function.

vironment have different maximum and minimum reward they can produce. Thus it is unfair to compute the absolute regret $r_{\max} - r_w$. We thus compute the relative regret, by first taking the worst cases reward $r_{\min}$ in each environment. Then the relative regret is $\text{regret}_w = (r_{\max} - r_w)/(r_{\max} - r_{\min}) = (r_{w^*} - r_w)/(r_{w^*} - r_{\min})$.

Experiment Results

We plot the performance of posterior mean and the proxy reward $\tilde{w}$ under different methods in Fig. 4.4. To mitigate the effect of the choice of initial environments, we experiment on six random initial environments and find that our active method outperforms the proxy reward $\tilde{w}$, as well as the random and difficult baseline in both joint and independent modes (p ¡ 0.05).

Our active method (orange) outperforms both the random baseline (grey) and the difficulty heuristic (blue) as shown in Fig. 4.4. This means that certain environments, despite being rated "easy" by heuristics, turn out useful for the reward design. These environments are often edge-cases that subtly trigger undesirable behaviors — in Fig. 4.2 a rather empty road with two human vehicles triggered the autonomous vehicle to crash. Next we further investigate this.

Edge-Case Nature of Proposed Environments To study why our method performs better than the baselines, we measure the quality of the proposed environments using: $r(M_{\text{next}}) = \mathbb{E}_{\tilde{w}\sim P(w=w^*)}\left[\text{Regret}(\tilde{w}; w^*, M_{\text{next}})\right] / \mathbb{E}_{\tilde{w}\sim P(w=w^*)}\mathbb{E}_{M\sim\mathcal{M}_{\text{deploy}}}\left[\text{Regret}(\tilde{w}; w^*, M)\right]$, the ratio of regret on next environment versus average regret in deployment. Higher $r(M_{\text{next}})$ means the next environment better uncovers the overall regret of the current

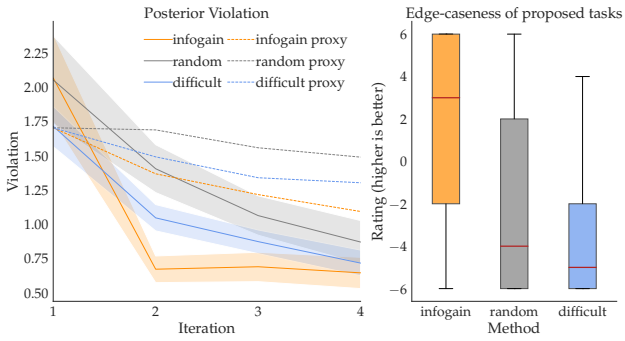


Figure 4.6: Experiment with robotic experts. The left shows the violation under different acquisition functions. The right compares the relative value of proposed environments.

posterior. As shown in Fig. 4.4, our active method proposes environments of highest relative regret. This suggests that to find the most useful environment for Assisted Reward Design, simply relying on heuristic environment ranking is not enough. It is more effective to utilize all proxy rewards and their induced uncertainty over w^*.

4.4 Experiments on Real Reward Design

We conducted a study with 14 robotics experts[4] on the simplified autonomous driving tasks, where we ask them to design reward functions to generate their preferred driving behaviors.

Experiment Setup

We give the experts 10 minutes to familiarize with the features. We then let them design reward functions over four consecutive iterations, under three different methods and three different random seeds. This gives a total of 36 individual reward designs. We use counterbalancing to randomize the sequence of different methods. All our experiments are conducted over Zoom.

Violation. Since we do not have the ground truth reward function, we cannot measure posterior regret as in Sec. 4.3. For evaluation purposes only, we introduce seven violation criteria to measure the quality of our reward design, including collision, driving off track, stopping, etc.

Violations. Here we provide in Chapter 4.2 the definition of environment constraint functions used in the experiment section to evaluate driving quality. Each constraint is a boolean function that returns true or false for one timestep, and we compute the final violation count for each trajectory by summing over constraints over the time horizon. These

[4] from the authors' institution

constraint functions are not used for optimization, but as an evaluation criterion for the case study experiment. Lower violation counts correspond to better driving behavior.

Environment Constraints		
Feature Name	Definition	Meaning
Overspeed	$v > v_{\max}$	The vehicle is driving over the maximal allowed speed.
Underspeed	$v < v_{\min}$	The vehicle is driving below the minimum speed on highway (i.e. backing up).
Uncomfortable	$\|\|u\|\|_\infty > \|\|u_{\max}\|\|_\infty$	The vehicle is applying too much force that it's uncomfortable (i.e. accerlating too much or jerky).
Collision	$\|\|\vec{x} - \vec{x}_{\text{car i}}\|\| \leq d_{\min}$	The vehicle crashes into the other vehicles.
Crash Object	$\|\|\vec{x} - \vec{x}_{\text{obj i}}\|\| \leq d_{\min}$	The vehicle crashes into the obstacles.
Offtrack	$x < x_{\text{fence left}}$ or $x > x_{\text{fence right}}$	The vehicle drives off the left or the right fence.
Wronglane	$\|\|x - x_{\text{lane left}}\|\| \leq d_{\text{lane}}$	The vehicle drives drives onto the wrong lane while merging.

Table 4.2: Environment Constraints Table

We use these criteria as unit tests on the behavior to capture unwanted behaviours. We then count the total number of violations of each trajectory. Better reward functions should induce lower violation counts. Although we could explicitly incorporate these tests as hard constraints in the "reward" function [2], we keep the reward features oblivious to them. This is for two main reasons. First, in the real world, constraint violations are acceptable in some edge-case scenarios: slamming on the brake, despite uncomfortable, is justifiable at the sight of traffic accidents. Such trade-offs are difficult to specify and it is our goal to exploit potential edge-cases. Second, violation tests only capture a subset of unwanted behaviors. Tailgating, for instance, does not break the tests. Yet, it is undesirable in most cases for the autonomous vehicle. In Sec. 4.4 we provide qualitative examples from the experiment to illustrate such undesirable behaviors.

Experiment Results

Evaluation. We perform Assisted Reward Design where experts answer the queries stemming from our algorithm, as well as the baselines. We visualize the result in Fig. 4.6 left.

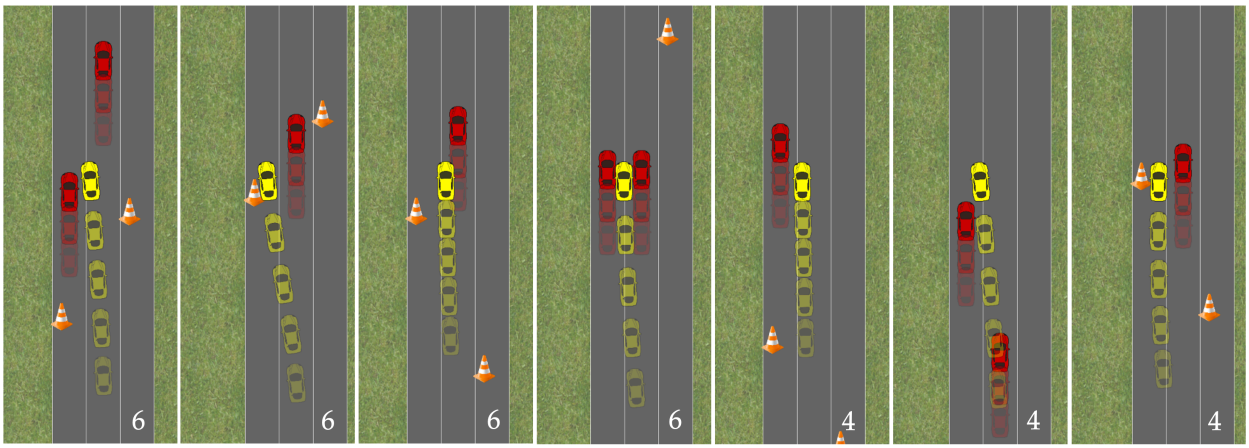


Figure 4.7: Selected environments proposed by Maximal Information with the highest rating by users. The robot plans a trajectory using previously designed w_{MAP}. Normalized Likert scale rating is shown on the bottom right of each scenario. Higher rating means more likely to be an edge case. Notice that these trajectories are failure cases but do not violate the pre-defined difficulty metrics such as collision.

Maximal Information outperforms the other acquisition functions in causing fewest average violations on $\mathcal{M}_{\text{deploy}}$. This shows that Assisted Reward Design can help us tune a reward function that fails fewer tests as a measurable proxy for the good driving behaviors in the real world.

Edge-Case Nature of Proposed Environments. To measure the quality of the proposed environments at each iteration, we ask the experts to watch the trajectories that their proxy designs generate on the proposed tasks. We then use the Likert scale questions in Fig. 4.5 and ask the users to rate how much they agree with the statement on a scale of 1-7. We then compute the normalized score $Q1 - Q2$ as a measure of how good the behavior is. A lower score indicates better behavior. We use the inverse of the score to indicate edge-caseness in Fig. 4.6.

Our maximal information method outperforms the heuristic difficulty metric in finding edge-case environments that contain more violations.

Qualitative Analysis of Proposed Environments We randomly sample an initial environment and query designer. We then use maximal information, heuristic difficulty, and random acquisition function to propose the next environments. We visualize the trajectory of MAP estimate w_{MAP} on each of these new environments in Sec. 4.4. Notice that interestingly, our method tends to find environments where the current MAP estimate fails, even though we do not explicitly optimize for finding failure cases. It turns out that these environments help reward designers effectively narrow down on the true reward. In comparison, heuristic difficulty finds environments with dense interactions. Yet, they do not induce failure in the resulting trajectories likely because such difficulties have

been addressed by past reward designs. Heuristic difficulty tends to generate repetitive environments.

Can Violation Count Replace the Maximal Information? Sec. 4.4 uses violation counts to evaluate trajectories. One may ask the question: can we simply use the violation count as an acquisition function to find edge-case environments? In other words, environments where the design leads to high collision, off-track counts, etc. should be proposed as edge cases. On a surface level, this seems to be a reasonable approach. In fact, this is commonly used in industry [181]. However, we argue that setting violation counts as the acquisition function limits the scope that we can identify edge cases. We highlight this in Fig. 4.7, where we visualize selected environments proposed by Maximal Information. They exhibit subtle failures and lead to the highest Likert-scale ratings despite obeying all violation criteria. Interestingly in the second, fourth, and seventh examples, the vehicle narrowly avoids the collision. Such behaviors found by Maximal Information are hard to capture with hand-designed violation metrics. By acting on the edge of violations, the algorithm pushes the reward designer to better specify her "decision boundaries". As a result, Maximal Information can propose more diverse edge cases and gather designer preference more quickly.

4.5 Conclusion

We contribute Assisted Reward Design, an active reward learning method that generates environments and queries for reward functions as user inputs. We find experimentally that our method exposes edge cases — environments where the current proxy reward leads to high regret. Experiments with robotic experts confirmed that these environments are edge cases that lead to undesirable behaviors.

Our method is applicable to model predictive control problems such as drone, indoor robot navigation, motion planning for manipulation [147] where we have known dynamics, as well as RL problems given the environment simulator or learned model. Given that naively applying RL in the inner loop of reward inference is intractable, it remains an open question how to efficiently combine with RL.

There are several limitations to our approach. Firstly, our environment proposal algorithm could be more efficient. One may explore synthesis and continuous optimization over the environment parameters. Secondly, we need of formal analysis and guarantee of generating edge cases. Thirdly, our method assumes access to a distribution of environments. Obtaining such realistic distributions [44, 205, 78] for simulation remain an open problem for the community.

Part II

Robustness in Interaction

Chapter 5

Test-time Adaptation for Assistive Robotics

Despite our effort at training and development time, we inevitably run into out-of-distribution humans at test time. This chapter explores how we can make the best use of training to enable test-time adaptation.

5.1 Why do we need test-time adaptation?

Our ultimate goal is to enable robots to assist people with day to day tasks. In the context of patients with motor impairments, this might mean assistance with scratching an itch, bathing, or dressing [120, 20, 47]. These are tasks in which doing reinforcement learning from scratch in the real world is not feasible, and so sim2real transfer is an appealing avenue of research. Sim2real methods for physical robot tasks in isolation typically work by constructing a diverse ”population” of environments and training policies that can work with any member of the population (e.g. a range of parameters of a physics simulator or a range of lighting and textures) [184, 180, 206, 95, 112, 200, 162, 70].

Similarly, population-based (self-play) training has proven successful in zero-sum games against humans [166, 9, 79]. But unlike tasks the robot does in isolation, assistance requires coordinating with a human who is also acting. And unlike competitive settings, assuming the human to be optimal when they are not, can result in dramatically poor performance [29]. Thus, in sim2real for assistance, we have to design a population of potential users and strategies to train with, akin to the physical environment parameters in typical sim2real tasks, rather than the standard population-based training approaches used in competitive settings. But designing a population that is diverse and useful enough to enable generalization to test-time humans, each with their own preferences, strategies, and capabilities, remains very challenging, making it likely that test-time partners might lie outside of the distribution the population was drawn from. Therefore, sim2real methods for assistance will need to be ready to *generalize* to *out-of-distribution* partner policies.

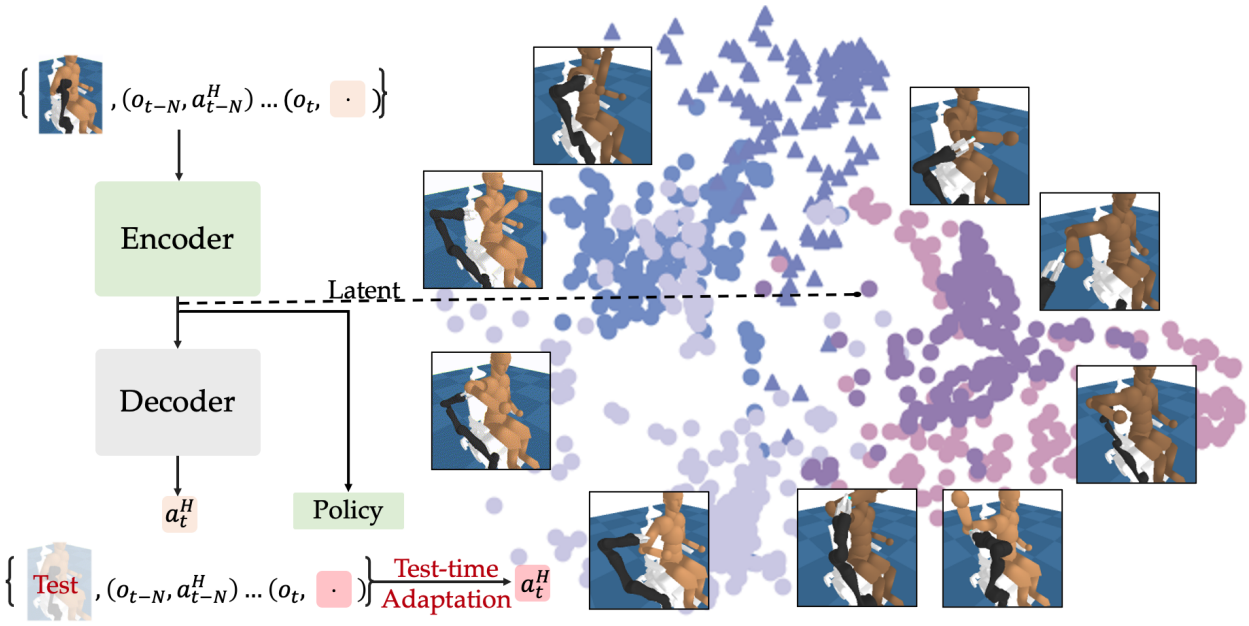


Figure 5.1: The framework for jointly learning the Personalized Latent Embedding Space and the robot policy. During training time, we train all components end-to-end to optimize for action prediction (orange) and the robot policy (green). At test time, we can further optimize for this objective to perform test-time adaptation (red). The resulting latent space captures the underlying structure of the preferences and strategies of the training humans.

In this work, we identify two principles as key to enabling better generalization. First is that we benefit from learning a latent space of partners that distills their policies down to a structure that is useful for the robot's policy *and* that makes it easy to identify partners at test time. Second is that we need to be prepared for this space to not perfectly capture the space of real human policies, and design it so that it is *adaptable* with real test-time interaction data.

We thus propose a framework that *learns a latent space directly from history of interaction by predicting the partner's actions*. Our framework allows a robot to capture the relevant information about the human partner that the robot can actually identify when starting to interact, and also enables test-time adaptation of the latent space itself when observing the partner's actions. When evaluated with partner policies we purposefully design to be out-of-distribution, we find that our approach leads to better generalization than prior methods which either do not learn a latent space at all [173], do not learn a latent space directly based on interaction history [95], or train a latent space based on other observables, like states or rewards [203, 135]. Our contributions are four-fold:

1. We introduce an assistive problem setting where the focus is explicitly on general-

ization to out-of-distribution partner policies.

2. We introduce a framework for training policies for this problem setting, Prediction-based Assistive Latent eMbedding (PALM). This enables us to study different methods for learning latent representations on how well they enable generalization.

3. We identify that the design choice of training a latent space by *predicting partner actions directly from history* outperforms (1) state-of-the-art sim2real approaches used in non-assistive tasks that are based on embedding environment parameters [95] as well as (2) human-robot interaction approaches that train representations by predicting observed states or rewards [203, 135].

4. We propose to adapt the learned latent space at test time, upon observing the partner's actions, and show it leads to generalization performance gains.

5.2 The Assistive Personalization Problem

In this section, we introduce the personalization problem in an assistive context. In particular, our goal is to learn a robot policy π_R that can assist a novel human partner in zero-shot fashion, or with a small amount of test-time data.

Two-player Dec-POMDP. An assistive task can be modeled as a two-agent, finite horizon decentralized partially-observable Markov decision process (Dec-POMDP) and is defined by a tuple $\langle S, \alpha, A_R, A_H, \mathcal{T}, \Omega_R, \Omega_H, O, R \rangle$. Here S is the state space and A_R, A_H are the human's and the robot's action spaces, respectively. The human and the robot share a real-valued reward function $R : S \times A_R \times A_H \to \mathbb{R}$; however, we assume that the reward function is not necessarily observed by the robot, i.e. its parameters (e.g. the location of the human has an itch) are in the hidden part of the state. $\mathcal{T} : S \times A_R \times A_H \times S \to [0, 1]$ is the transition function, which outputs the probability of the next state given the current state and all agents' actions. Ω_R and Ω_H are the sets of observations for the robot and human, respectively, and $O : S \times A_R \times A_H \to \Omega_R \times \Omega_H$ represents the observation probabilities. We denote the horizon of the MDP by T.

Target User. We target users with partial motor functions — a common impairment for individuals with partial arm functions. This is an impairment that can occur in some people with cervical SCI, ALS, MS, and some neurodegenerative diseases — leading to the need for robotic assistance. We model the extent of the impairment as the privileged information in the Dec-POMDP. The robot does not know this a-priori and thus needs to adapt to individual users' capabilities.

The Robotic Caregiving Setup. We define the observation space for the robot and the human following [47]: the robot observes its own joint angles, and the human's joint positions in the world coordinate and contact forces; the human observes their joint angles (proprioception) and the end-effector position of the robot. When training with simulated humans, the robot gets a reward signal (which depends on privileged information), and

has to use that signal to learn to implicitly identify enough about the human to be useful; at test time, the robot does not observe reward signal and must use what it has learned at training time to identify the human's privileged information and be helpful.

Distributions of Humans. Let function $\pi_H : \Omega_H^* \times A_H \to [0, 1]$ be the human policy that maps from local histories of observations $\mathbf{o}_t^H = (o_1^H, \ldots, o_t^H)$ over Ω_H to actions. We define two distributions of human policies $\pi_H \in \mathscr{D}_{\text{train}}, \mathscr{D}_{\text{test}}$. In the assistive itch scratching, $\mathscr{D}_{\text{train}}$ can be a set of humans with different itch positions on their arms, which lead to their different movements. We refer to them as *in-distribution humans*. $\mathscr{D}_{\text{test}}$ contains *out-of-distribution* humans whose itch position differ from those in the $\mathscr{D}_{\text{train}}$. At training time, the robot has access to $\mathscr{D}_{\text{train}}$. Thus, it has ground-truth knowledge about the training human's privileged information, such as each human's itch position. At test time, we evaluate the robot policy by sampling humans $\pi_H \sim \mathscr{D}_{\text{test}}$ and directly pairing them with the robot policy. We evaluate the zero-shot and few-shot adaptation performance of the robot policy.

Objective. The main problem we study is how to leverage the training distribution to learn a robot policy $\pi_R : \Omega_R^* \to A_R$ such that we achieve the best performance on test humans. Concretely, we define the performance of the robot and human as

$$J(\pi_R, \pi_H) = \mathbb{E}\left[\sum_{t=0}^{T} R(s_t, \pi_R(\mathbf{o}_t^R), \pi_H(\mathbf{o}_t^H))\right], \tag{5.1}$$

Only given access to $\mathscr{D}_{\text{train}}$, our objective is to find the robot policy
$\pi_R = \arg\max_\pi J(\pi, \pi_H), \pi_H \sim \mathscr{D}_{\text{test}}$.

5.3 Learning Personalized Embeddings for Assistance with PALM

In this section, we present Prediction-based Assistive Latent eMbedding (PALM). We introduce the general framework of using a latent space to perform personalization in an assistive context. We then highlight the advantage of action prediction in contrast to prior works. Finally, we describe how we can optimize PALM at test time to work with out-of-distribution humans.

Learning an Assistive Latent Space

Given a training distribution of humans $\mathscr{D}_{\text{train}}$ [1], we would like to learn a robot policy that can adapt to assist new users. To achieve that, a robot must learn to solve the task while efficiently inferring the hidden component that differs across humans. One natural way to do so is to *learn a latent space* that succinctly captures what differs across humans in a

[1] we describe how we generate this distribution in Sec. 5.4

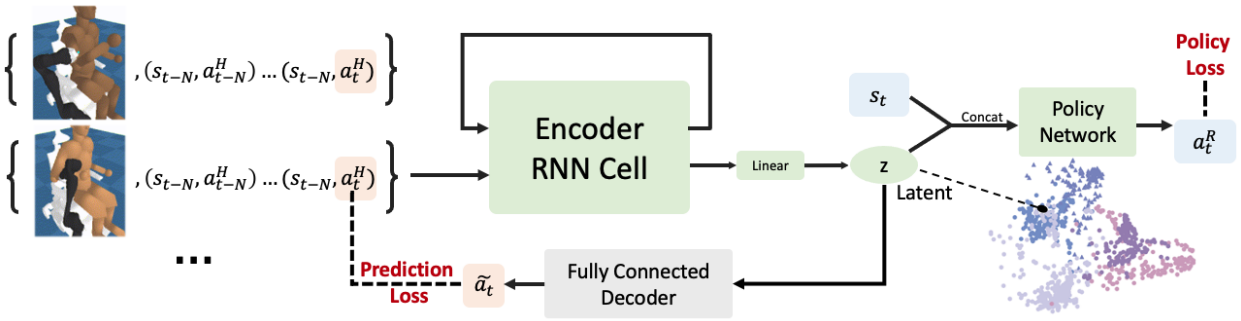


Figure 5.2: The framework for jointly learning the Personalized Latent Embedding Space and the robot policy. During training time, we train all components end-to-end to optimize the prediction loss and the policy loss. At test time, we execute the robot policy, which is highlighted in green.

way that affects the robot's policy. When deployed on a test human, the robot infers this latent embedding and uses it to generate personalized assistance.

We denote the latent space as $z_t \sim \mathcal{E}_\theta(z; \tau_{1:t})$, where $\mathcal{E}_\theta$ encodes the trajectory τ so far and outputs latent vector z_t. The robot uses this latent space to compute its actions $a^R_t = \pi_R(o^R_t, z_t)$. We train the base policy π_R and the latent space encoder $\mathcal{E}_\theta$ jointly as they are interdependent [107] — better robot policy leads to different trajectories across humans, which in turn leads to distinguishing z. Ideally, we would like z to capture sufficient information to differentiate the humans, similar to performing a "dimensionality reduction" on human policies π_H. We hereby introduce different objectives for learning such latent space, and how our method — learning by action prediction — makes a good fit for the assistive personalization problem.

We do this by dividing the problem into learning a human embedding z for every user and a common robot policy network $a \sim \pi_R(\cdot|o_R, z)$ for generating actions. While the robot policy network π_R is shared among different users, the embedding z is extracted from human motion trajectories. Under this construct, a good personalized embedding space should capture the most important attributes of human policies that differ, and pass over the common components shared by all humans — these commonalities can be handled by the base policy. In other words, our goal is to perform *dimensionality reduction* on all human policies that we have in the training distribution.

A good personalized embedding space should capture the most important attributes of human policies that differ, and pay less attention to the common components shared by all humans — these commonalities can be handled by the robot base policy, and do not require personalization.

Additional Training and Implementation Details

In our experiments in both the assistive reaching and assistive itch scratching, we use a recurrent network over a sliding window of 4-time steps, each of which is a concatenated vector of observation o_t, human action $a_{\mathrm{H}t-i}$ and robot actions $a_{\mathrm{R}t-i}$. For the current time step t, we use zero vector for $a_{\mathrm{H}t}$ and $a_{\mathrm{R}t}$. We set the latent space dimension to be four, and use a recurrent network with six layers. Our base policy network has four dimensions and hidden size of 100.

How to Construct the Latent Space

Prior work and limitations LILI [203] and RILI [135] learn a latent embedding of the humans by predicting the next observations and rewards. While they have been shown to work in predicting and influencing human behaviours, both methods assume access to the reward function at test time, which we do not have access to in the assistive setting — we don't know a-priori the preference and needs of a new user. RMA [95] enables fast robot adaptation by learning a latent space of environment parameters, such as friction, payload, terrain type, etc. While it works for a single robot, it is unclear in human-robot settings, how we can encode human motions and preferences as environment parameters.

Learning by action prediction. Given history $\tau_{t-N:t} = \big((o^R_{t-N}, a^H_{t-N}), \dots (o^R_t, \cdot)\big)$ of N robot observation and human action[2] pairs, as outlined in Fig. 5.2, we embed this trajectory to a low-dimensional manifold and use it to predict a^H_t. The intuition is that if we are able to predict the next action by this human, we extract the sufficient information about the human's policy π_H. The latent vector z is representative of the trajectory so far and indicative of the person's future actions. We do this by training a decoder $\mathcal{D}_\phi$ parameterized by ϕ to predict the next action from the encoder's output $z \sim \mathcal{E}_\theta(z; \tau_{1:t})$.

$$\mathcal{L}_{\text{pred}} = \min_{\theta,\phi} ||\mathcal{D}_\phi(z) - a^H_{t+1}||^2 + c_{\text{KL}} \cdot KL(\mathcal{E}_\theta(z; \tau_{1:t})||\mathcal{N}(z)) \tag{5.2}$$

The encoder $\mathcal{E}$ is a recurrent neural network parameterized by θ. Here the second term is a regularization term motivated by Variational Autoencoder [87, 197], that enforces the latent space to follow a normal prior distribution. This encourages nearby terms in the latent space to encode similar semantic meanings. In the context of assistive tasks, this helps us better cluster similar humans closer in the latent space, and we show a didactic example in Sec. 5.4.

Latent Space Adaptation at Test Time

At test time, as we work with a new user, we would like our encoding of the new user to match the true latent information, z^* of that user. In other words, we would like to

[2]We do not assume access to the person's sensorimotor action (e.g. joint torques). We define human action as change in the person's Cartesian pose, which can be tracked externally.

minimize $||\mathcal{E}(\tau) - z^*||^2$. Because we do not know about the new users a-priori, we can only optimize for this objective via proxy, which we refer to as *test time adaptation*.

Since the PALM latent space is based on action prediction, we can adapt it to a new user by further optimizing the latent space. Note that because Eq. (5.2) requires only observation-action data, we do not need any additional label to perform test time adaptation. More formally, we collect a small dataset of test-time interaction trajectories, τ, and perform a few gradient steps to optimize both the encoder and decoder for Eq. (5.2): $(\theta, \phi) \rightarrow (\theta, \phi) - \delta \nabla_{(\theta,\phi)} \mathcal{L}_{\text{pred}}(\mathcal{E}_\theta, \mathcal{D}_\phi, \tau)$.

The idea of test-time optimization has been shown to improve perceptual robustness for grasping in sim2real research [204]. We follow a similar pipeline, where we can improve the latent encoding by collecting unsupervised action data from test users. Here the main difference is instead of perceptual differences, our goal is to reduce the domain gap on test users.

End-to-end training of the latent space

We need to train the base policy π_R and the latent space encoder $\mathcal{E}_\theta(z; \tau)$ jointly because they are interdependent — z is the input to π_R, and π_R decides the data distribution which leads to z. We simultaneously optimize the prediction loss in Eq. (5.2) and the policy loss using PPO [160] algorithm. See Sec. 5.3 for more training details. To amend for the instability of training with a population of humans, we leverage Behaviour Cloning, where we use expert robot policies obtained via co-optimization to supervise π_R on on-policy data.

More specifically, we query the expert actions a_t^{exp} in a DAgger fashion[150] [3] during training, and optimize a_t^R to minimize deviation from it: $\mathcal{L}_{\text{BC}} = \sum_t ||a_t^{\text{exp}} - a_t^R||^2$. The overall policy optimization loss include three terms: latent prediction loss, PPO loss and the Behaviour Cloning loss: $\mathcal{L}_{PALM} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{PPO}} + \mathcal{L}_{\text{BC}}$.

Generating Human Populations

To train our robot using sim2real, we would like to have a set of diverse environments. However, unlike single-agent domain randomization where we can vary environment parameters such as friction, in assistive tasks the environment entails a changing user policy. It is not obvious how to best generate a diverse population that captures user preferences, levels of disabilities, or movement characteristics. Generating human motions that realistically capture the variation observed in physical human-robot interaction has remained an unsolved challenge in robotics.

Co-Optimization Prior works in robotic assistance [47, 36] have demonstrated that by optimizing for the same task objective, we can generate human and robot motions that

[3] This ensures that we encounter no distribution shift at deployment time.

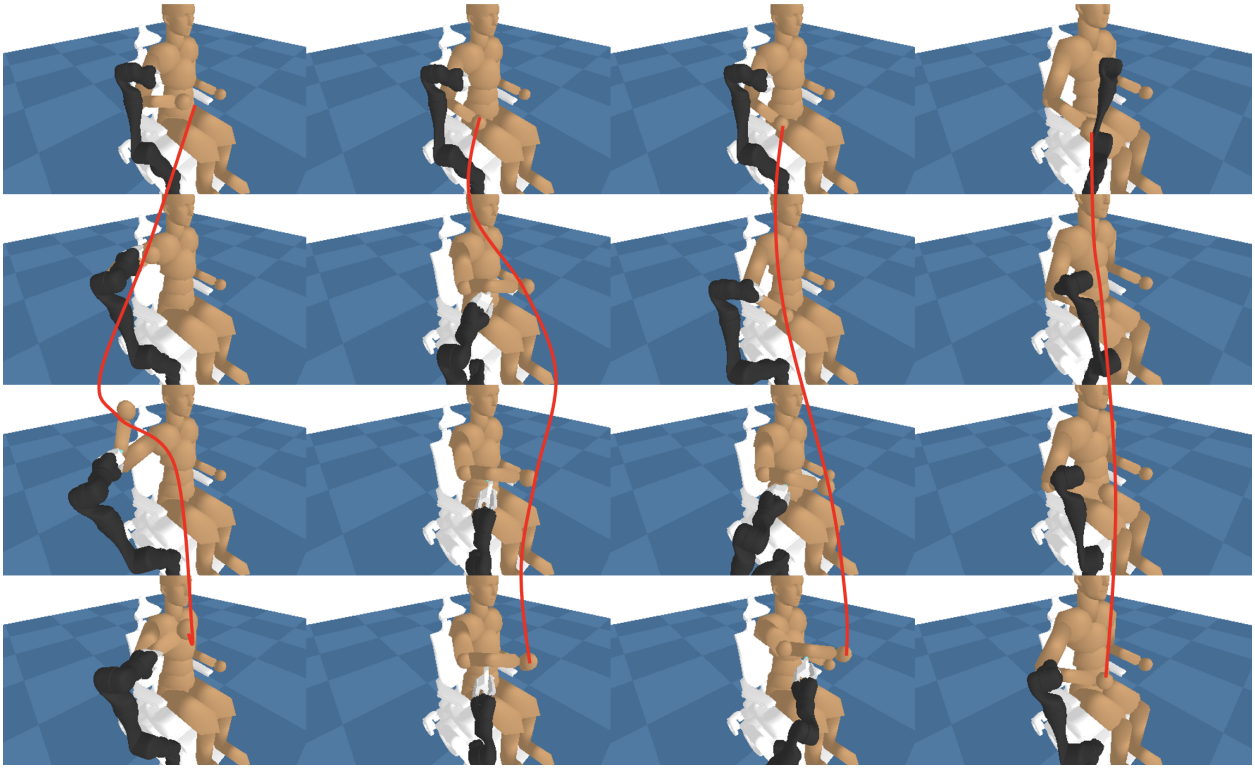


Figure 5.3: Visualization of humans generated of different activity levels. From left to right we apply action penalties $c_p = 0, 10, 30, 60$. Qualitatively, increasing the penalty results in the human taking more steady actions with less swinging motions. This results in the human being more likely to expose the itch spot for the robot to scratch, as opposed to scratching themselves.

coordinate towards the same goal, such as robot-assisted dressing. Further more, we can leverage reward engineering [61, 210] to generate a diverse set of motions.

To generate diverse population in itch scratching task, we explore two sources of diversities: (1) we assign different human action penalty c_p, where larger penalties lead to the human agent exerting less effort. In the simulation experiment we use $c_p = 3, 3.5, 4$.

(2) We simulate different itch positions on the human's arm and train co-optimized human and robot policies conditioned on them. This leads to qualitatively different strategies for the human and the robot. Note that this serves as the first step to understanding how different methods generalize since we never expect to be able to capture the diversity in humans perfectly.

For training, we use Proximal Policy Optimization (PPO) to optimize human and robot policies in an interleaving fashion.

Note that we also keep the co-optimized robot policy and use it to obtain expert actions for assistive policy training (see Sec. 5.3 and supplement for full details).

Visualization Here we visualize trajectories from humans with different action penalties in Fig. 5.12. Note that higher penalties result in the human taking more steady actions of smaller magnitude.

Algorithm

We present the main algorithm for PALM assume we have access to training and test distributions $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$.

Algorithm 3 Prediction-based Assistive Latent eMbedding Training

Randomly initialize base policy π, encoder $\mathcal{E}_\phi$ parameterized by ϕ, decoder $\mathcal{D}_\theta$ parameterized by θ. Empty replay buffer D_1, window size w

for itr $= 1, .., N_{\text{itr}}$ **do**
  for $i = 1, .., N_{\text{batch}}$ **do** Sample $\pi_{\text{H}_i} \sim \mathscr{D}_{\text{train}}$, optionally find pre-trained expert robot policy $\pi_{\text{E}i}$
    $o_0 \leftarrow$ env.reset()
    Initialize history $H \leftarrow \phi$
    for $t = 0, ..., T$ **do**
    Get latest w steps from H: $H_{-w} \leftarrow H[-w:]$.
    $z_t \leftarrow \mathcal{E}_\phi(o_t, H_{-w})$
    $a_{\text{H}t} \leftarrow \pi_{\text{H}_i}(o_0)$
    $a_{\text{R}t} \leftarrow \pi(o_t, z_t), a_{\text{e}t} \leftarrow \pi_{\text{E}i}(o_t)$
    $o_{t+1} \leftarrow$ env.step$(a_{\text{H}t}, a_{\text{R}t})$
    Store $(o_t, a_{\text{H}t}, a_{\text{R}t}, a_{\text{e}t}, H_{-w})$ in D_1
    end for
  end for
  for $j = 1, ..., N_{\text{opt}}$ **do** Sample a batch of $(o_t, a_{\text{H}t}, a_{\text{R}t}, H_{-\tau})$ from D_1
    Compute $\mathcal{L}_{\text{pred}}$ using Eq. (5.2), $\mathcal{L}_{\text{PPO}}$ using [160] and $\mathcal{L}_{\text{BC}} = \sum_t ||a_t^{\text{exp}} - a_t^R||^2$
    Optimize θ, ϕ, π for $\mathcal{L}_{\text{PALM}} = \lambda_{\text{pred}}\mathcal{L}_{\text{pred}} + \lambda_{\text{PPO}}\mathcal{L}_{\text{PPO}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}$
  end for
end for

5.4 Experiments

In this section, we evaluate our method PALM (Prediction-based Assistive Latent eMbedding) in collaborative human-robot environments of varying tasks and varying populations of human models. In particular, we focus on the out-of-distribution generalization by constructing different forms of out-of-distribution populations. We focus on empirically investigating the benefits of learning a latent space, the effect of different kinds of

Algorithm 4 Prediction-based Assistive Latent eMbedding Test Time Adaptation

Sample $\pi_{\mathrm{H}} \sim \mathscr{D}_{\text{test}}$, Initialize history $H \leftarrow \phi$, Empty trajectory data τ
for $i = 0, .., N_{\text{adapt}}$ **do**
  $o_0 \leftarrow$ env.reset()
  for $t = 0, ..., T$ **do**
    Get latest w steps from H: $H_{-w} \leftarrow H[-w:]$.
    $z_t \leftarrow \mathcal{E}_\phi(o_t, H_{-w})$
    $a_{\mathrm{R}t} \leftarrow \pi(o_t, z_t)$
    $o_{t+1} \leftarrow$ env.step$(a_{\mathrm{H}t}, a_{\mathrm{R}t})$
    Store $(o_t, a_{\mathrm{H}t}, H_{-w})$ in τ
  end for
    Compute $\mathcal{L}_{\text{pred}}$ using Eq. (5.2) on τ, $\theta \rightarrow \theta - \delta \nabla_\theta \mathcal{L}_{\text{pred}}(\mathcal{E}_\theta, \tau)$.
end for

prediction on learning a useful latent space, the properties of learned latent spaces, and the gains from test-time adaptation to humans.

Environments

Here we introduce two environments where we study assistive personalization. In both environments, the robot has to infer some hidden information from the human in order to successfully solve the task. Note that these are examples meant for demonstrating the effectiveness of the algorithm, and we do not claim to solve the full robotic caregiving problem.

Assistive Reacher (Fig. 5.4) is 2D collaborative environment where a two-link robot arm assists a point human agent to get to the target position. This target is located at $(d\cos\alpha_{\mathrm{H}}, d\sin\alpha_{\mathrm{H}})$, where d is a fixed value, and $\alpha_{\mathrm{H}} \in [-\pi, \pi]$ is known to the human, but not the robot. The human agent is initialized randomly in the 2D plane with random hidden parameters $\alpha_{\mathrm{H}} \in [-\pi, \pi], k_{\mathrm{H}} \in [0.5, 1.5]$. The robot can only identify the target position by physical interactions — once the robot initiates contact, the human applies a force $k_{\mathrm{H}} \cdot (\cos\alpha_{\mathrm{H}} + \frac{\pi}{2}, \sin\alpha_{\mathrm{H}} + \frac{\pi}{2})$. Only by recognizing the human in terms of $\alpha_{\mathrm{H}}, k_{\mathrm{H}}$ can the robot compensate the force, and successfully move the human to the hidden target. Each episode has 40 timesteps.

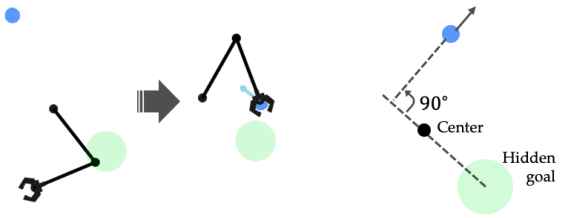


Figure 5.4: The assistive reacher environment. Left: the robot's goal is to move the human agent towards the hidden target. Right: the hidden goal's position can be inferred 90 degrees from the humans force output.

The Scope of Generalization. We define $\mathscr{D}_{\text{train}}$ as 36 samples uniformly sampled from $\alpha_{\text{H}} \in [-\pi, \frac{\pi}{2}], k_{\text{H}} \in [0.5, 1.5]$ and $\mathscr{D}_{\text{test}}$ as 12 samples uniformly sampled from $\alpha_{\text{H}} \in [\frac{\pi}{2}, \pi], k_{\text{H}} \in [0.5, 1.5]$.

Assistive Itch Scratching (Fig. 5.2) is adapted from assistive gym [47]. It consists of a human and a wheelchair-mounted 7-dof Jaco robot arm. The human has limited mobility — they can only move the 10 joints on the right arm and upper chest, and needs the robot's assistance to scratch the itch. An itch spot is randomly generated on the human's right arm. The robot does not directly observe the itch spot, and relies on interaction with the human to infer its location. Each episode has 100 timesteps.

Generating Synthetic Human Agents

We use co-optimization to create $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$ for Assistive Itch Scratching.

A benefit of the co-optimization framework is that it naturally induces reward-seeking behaviour from the human and the robot, which simulates assistance scenarios. For instance, to generate more inactive human policies, we can introduce a weighting term in the reward function for human action penalties $R_{\text{p}} = c_{\text{p}} \cdot ||\pi_{\text{H}}(s_t)||^2$ where c_{p} is a constant controlling the penalty. The overall objective becomes

$$\max_{\pi_{\text{H}}, \pi_{\text{R}}} \mathbb{E}\big[\sum_t R\big(s_t, \pi_{\text{H}}(s_t), \pi_{\text{R}}(s_t)\big)\big] + c_{\text{p}} \cdot ||\pi_{\text{H}}(s_t)||^2 \tag{5.3}$$

The Scope of Generalization. We are motivated by real world applications where users tend to have different levels of mobility limitations, or itch locations in different body parts. To generate a synthetic population to capture such diversity in itch scratching task, we explore different co-optimization settings (1) we assign different human action penalty to be $c_{\text{p}} = 3, 3.5, 4$, where larger penalties lead to the human agent exerting less effort. To compensate for the penalty, the human agent tends to move joints earlier in the kinematic chain of the arm, such as the shoulder instead of the elbow, which can induce larger motions. As a result, the varying penalties lead to qualitatively different solutions for the co-optimization. (2) We simulate different itch positions on the human's arm and train co-optimized human and robot policies conditioned on them. This leads to qualitatively different strategies for the human and the robot. Note that this serves first step to understanding how different methods generalize, since we never expect to be able to capture the diversity in humans perfectly. Indeed, figuring out a good synthetic population that can effectively model real-world humans is an important open question. Note that there are a variety of other different ways to generating synthetic humans and we do not claim that these are the most optimal way to generate them. Figuring out how to optimally generate synthetic human to achieve best real world performance remains an open challenge.

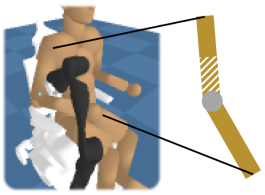


Figure 5.5: Definition of $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$.

For training, we use Proximal Policy Optimization (PPO) to optimize human and robot policies in an interleaving fashion.

Note that we also keep the co-optimized robot policy and use it to obtain expert actions for assistive policy training (see Sec. 5.3 and supplement for full details).

To construct $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$, we divide the two arms' areas into four equal portions, as shown in Fig. 5.5, and generate human policies conditioned on itch positions in these areas. $\mathscr{D}_{\text{train}}$ consists of three of the four portions and $\mathscr{D}_{\text{test}}$ consists of the remaining one. We then construct three distribution sets in increasing order of difficulties. In the first distribution $\mathscr{D}^1$, we confine itch positions from a line-shaped region. In $\mathscr{D}^2$, we sample from all the arm areas. Note that $\mathscr{D}^1$ and $\mathscr{D}^2$ are constructed by setting action penalty $c_{\text{p}} = 3$. In $\mathscr{D}^3$, we combine humans of $c_{\text{p}} = 3, 3.5, 4$, each trained with two different random seeds. This adds the extra complexity of human activity levels. We simulate 12 in-distribution humans from each of the three training portions under each action penalty.

Baselines

We compare with baselines that do not learn an explicit latent space as well as existing methods for adaptation via learning latent embeddings.

MLP and RNN. We follow [173] that trains sequential models to enable adaptation to simulated humans. We explore using a recurrent neural network or a feed-forward network on concatenated state-action histories. The models directly output robot action, and there is no latent space modeling.

PPO and Behaviour Cloning

We need to train the base policy π_R and the latent space encoder $\mathcal{E}_\theta(z; \tau)$ jointly because they are interdependent — z is the input to π_R, and π_R decides the data distribution which leads to z. We simultaneously optimize the prediction loss in Eq. (5.2) and the policy loss using PPO [160] algorithm. See Sec. 5.3 for more training details. To amend for the instability of training with a population of humans, we leverage Behaviour Cloning, where we use expert robot policies obtained via co-optimization (see Sec. 5.3) to supervise π_R on on-policy data. More specifically, we query the expert actions a_t^{exp} in a DAgger fashion[150] [4] during training, and optimize a_t^R to minimize deviation from it: $\mathcal{L}_{\text{BC}} = \sum_t ||a_t^{\text{exp}} - a_t^R||^2$. The overall policy optimization loss include three terms: latent prediction loss, PPO loss and the Behaviour Cloning loss: $\mathcal{L}_{\text{PALM}} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{PPO}} + \mathcal{L}_{\text{BC}}$.

Hyperparameters

PALM Training We use $\lambda_{\text{PPO}} = 0.1, \lambda_{\text{BC}} = 1, \lambda_{\text{pred}} = 0.1$ in our experiments. We find that the behaviour cloning loss is essential for the Assistive Itch Scratching task. Under

[4] This ensures that we encounter no distribution shift at deployment time.

this hyperparameter setting, we train for 200 iterations. During each iteration, we collect 19,200 state-action transitions, which is evenly divided into 20 mini-batches. Each mini-batch is fed to the base policy and encoder for 30 rounds to compute the loss and error for back-propagation. We set the learning rate to be 0.00005.

For PALM prediction training, we use a three-layer decoder with hidden size 12 to predict the next human action from the hidden state from the encoder. To implement the KL regularization, follow the standard VAE approach. We use two linear networks to transform the encoder hidden state into μ and σ, which denote the mean and the standard deviation of the latent space. We then compute approximate KL divergence to normal distribution on this latent distribution.

PALM Test Time Optimization At test time, we roll out the trained robot policy and collect data with the same user for 25 iterations, or 2,500 time steps. This amounts to 150 seconds of wall clock time. We then optimize for the prediction loss (including KL regularization term) using learning rate of 0.0001 for one to five steps, and use the one with the lowest loss. We empirically find the hyperparameters by doing the same process with humans from the training distribution, where we collect a mini training set and mini evaluation set both of 25 iterations. We use the mini training set to find the learning rate and use the evaluation set to ensure there is no over-fitting.

RILI/LILI Training We follow a similar approach to PALM, except that we learn to predict the next state o_{t+1} and scalar reward.

RMA Training We follow the two-phase training procedure in [95]. Note that we find it crucial in phase 2 to train the encoder with on-policy data, meaning that the regression data is collected by rolling out actions output by the "recurrent learner", not the trained network from phase 1. The phase 1 network is used simply for generating labels.

RNN/MLP Training. For RNN, we directly feed the hidden state of the recurrent encoder to the policy network. The architectural difference between RNN and PALM is that we do not concatenate the current observation o_t to the encoded output. To ensure that the policy has at least the same capacity as PALM, we use a base policy with the same number of parameters as in PALM.

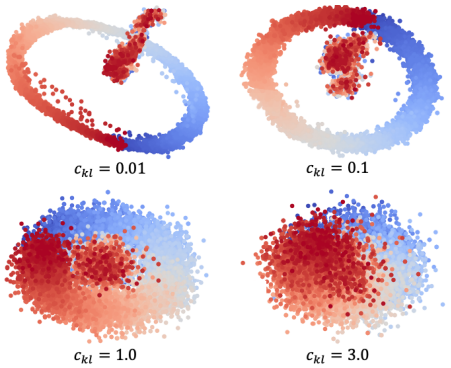


Figure 5.6: Latent space of PALM in the assistive reacher environment when we can sample humans $\alpha_H \in [-\pi \text{ (red)}, \pi \text{ (blue)}]$ continuously.

ID-based Human Embeddings. In contrast to learning latent space from history, another class of method studies encodes human-designed environment parameters [95, 107, 85]. We focus specifically on RMA [95], a two-phased method that first learns to encode task-ID (phase I) and then trains a recurrent network to regress to the embeddings from observation history (phase II). For training a quadruped robot, RMA encodes the physical parameters (friction, pay-

load, etc) of the environment. The first stage trains a policy with ground-truth information, and the second phase performs environment identification. While RMA is shown to be effective for learning policy for in-distribution environments, it is unclear how well it generalizes to out-of-distribution environments. Furthermore, in assistive tasks, it is unclear how to construct the "ground-truth ID" for phase I that quantifies the user characteristics. We study **RMA-Param** and **RMA-Onehot**, where we assign each training human a one-hot vector. For RMA-Param, we use a three-dimensional vector that includes the x, y position of the itch position and the arm index. When there are multiple human activity levels as mentioned in Sec. 5.4, we introduce a fourth dimension with an integer to indicate the action penalty c_{p}.

LILI and RILI. We consider two other methods of the PALM framework: LILI [203] and RILI [135]. As mentioned in Sec. 5.3, LILI jointly predicts future observation and reward, and RILI predicts reward. Given that reward is only available at training time, we cannot perform test time optimization for LILI and RILI.

Ablations of PALM. Our method has several components: we use a recurrent neural network to encode interaction history, and use its output to minimize prediction loss $\mathcal{L}_{\text{pred}}$ and policy loss $\mathcal{L}_{\text{pol}}$. We also use the KL term in Eq. (5.2) to regularize the human embeddings. To test the effectiveness of our method, we separate different parts and create a set of baselines. We hereby describe them in detail: (1) No $\mathcal{L}_{\text{pred}}$: the model shares the same encoder and policy network architecture, yet we don't optimize for $\mathcal{L}_{\text{pred}}$. By removing the prediction loss in Fig. 5.2, the latent space is not explicitly trained to contain human information. (2) No c_{KL}: no regularization in the latent space. (3) Frozen embedding: instead of jointly training embeddings and the policy network, we first train the encoder on expert data, freeze it, and then train robot policy.

Ablation Studies

PALM Baselines	Reach	Itch $\mathscr{D}^1$	Itch $\mathscr{D}^2$	Itch $\mathscr{D}^3$
PALM test optim	$\mathbf{0.43 \pm 0.13}$	$\mathbf{0.51 \pm 0.08}$	$\mathbf{0.50 \pm 0.02}$	$\mathbf{0.29 \pm 0.02}$
PALM w/o test optim	0.38 ± 0.10	0.48 ± 0.08	0.49 ± 0.02	0.26 ± 0.01
No $\mathcal{L}_{\text{pred}}$	0.30 ± 0.05	0.50 ± 0.08	0.45 ± 0.02	0.25 ± 0.01
$c_{\text{KL}} = 0$	0.32 ± 0.08	0.47 ± 0.04	0.46 ± 0.01	0.23 ± 0.02
Frozen $\mathcal{E}$	0.24 ± 0.04	0.21 ± 0.06	0.15 ± 0.07	0.11 ± 0.05

Table 5.1: Normalized Reward on $\mathscr{D}_{\text{test}}$, standard deviation over 3 seeds.

We include ablation studies of PALM in the main experiment in Sec. 5.4, where we study the effect of test-time optimization, prediction loss, KL regularization and jointly training encoder $\mathcal{E}$ and policy π.

In the assistive reaching experiment, we observe that test-time optimization, $\mathcal{L}_{\text{pred}}$, KL regularization, and joint training all contribute to the OOD performance.

In the assistive itch scratching experiment, test time optimization and $\mathcal{L}_{\text{pred}}$ improve experiment results in all the settings. Applying KL regularization provides some gain in the complex distribution $\mathscr{D}_3$, but does not lead to improvement in simpler distribution $\mathscr{D}_1$ and $\mathscr{D}_2$.

Didactic Experiment in Assistive Reacher Environment

Can PALM learn a meaningful distribution from the interaction? Unlike other ID-based methods like RMA, PALM does not have access to the human parameters at training time. We study whether PALM can learn a meaningful latent space without explicitly knowing this information. We sample training humans from $\alpha_{\text{H}} \in [-\pi, \pi]$ continuously. We train PALM with different amount of prior regularization, c_{KL} from Eq. (5.2). We train using a recurrent window of length 4 and a batch size of 512 episodes. Additional training details can be found in the supplementary material.

We average test results using 100 episodes and visualize the results in Fig. 5.4. Given that humans are parameterized by α_{H}, K_H, the ideal embedding space looks like a ring with a small blob in the center. The ring corresponds to the 2D projection of α_{H} and the blob denotes the initial part of interaction before contact, which is indistinguishable. We find that while PALM never observes the underlying parameter α_{H}, it can learn a latent space that characterizes α_{H}. Interestingly, varying the amount of regularization qualitatively affects the shape of the latent space. Setting the VAE regularization $c_{\text{KL}} = 0.1$ recovers a latent space that most resembles to the ideal latent space.

Assistive Reacher Main Experiment

Experiment Setting. We use the finite $\mathscr{D}_{\text{train}}$ described in Sec. 5.4 and train all baselines for 200 epochs with 512 batch size. We then evaluate the trained policies on $\mathscr{D}_{\text{test}}$. We normalize the resulting reward with respect to oracle reward.

Results. We average test results using 100 episodes. On in-distribution humans, we find that all methods successfully follow the right policy that assists the human to reach their goal. This shows that they all successfully predict the human latent information explicitly or implicitly. On out-of-distribution humans, the methods are no longer guaranteed to predict the correct embedding. PALM with action prediction significantly outperforms other methods. With test-time adaptation, PALM further improves.

Visualizing the latent space. We qualitatively study generalization by visualizing the latent space as well as the mapped embeddings of both in-distribution and out-of-distribution humans (in red crosses) in Fig. 5.7. Interestingly, only PALM with action prediction can infer the "ring" structure. RMA, RNN, LILI and RILI fail to do so. We hypothesize that because hand-crafted human IDs do not convey the information about human policy, RMA warped the IDs in arbitrary what that are harmful for generalization. The same happens with RILI and LILI. We hypothesize this is due to the inherent ambiguity in reward prediction: a low reward does not necessarily recover the human policy structure.

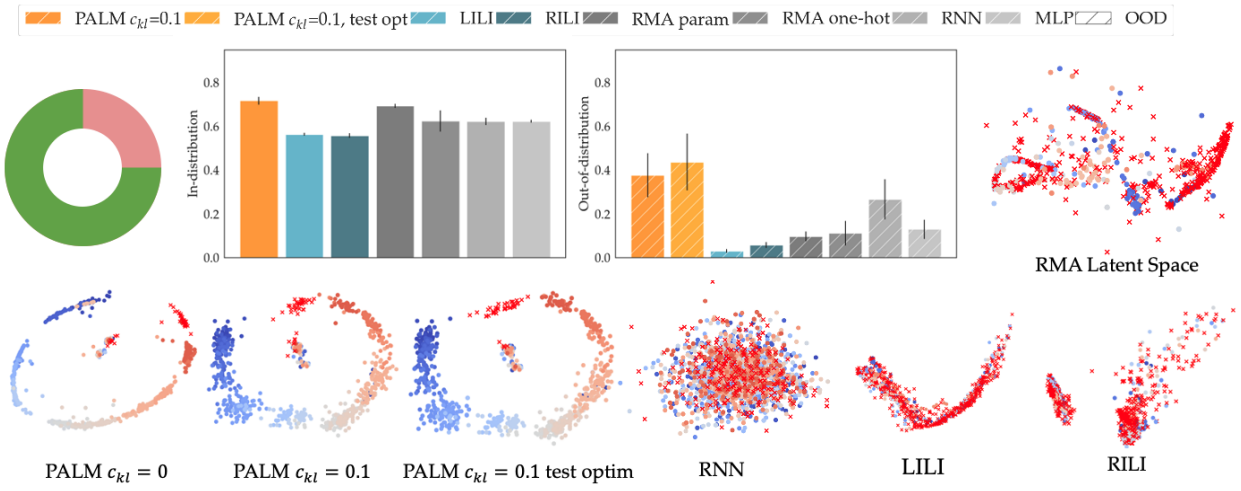


Figure 5.7: Top left: evaluation offi PALM and baselines on in-distribution (green) and out-of-distribution (pink) humans. Right and bottom: visualization of the latent embeddings of different methods. OOD humans are highlighted in red crosses. Best viewed electronically.

The visualization also offers some insight into why having c_{KL} regularization is helpful for generalization. Compare the latent space of PALM $c_{\text{KL}} = 0$ and PALM $c_{\text{KL}} = 0.1$, the latter induces a smoother distribution where test humans are better fitting in the "missing arc" of the "ring". Further more, we see that with test time optimization, the PALM latent space embeds the OOD human better, by filling in more of the arc.

Assistive Itch Scratch Main Experiment

Diverse population of humans. In the assistive gym environment, we followed the method in Sec. 5.4 and explore two axes of generating diverse synthetic humans. First, we use motion synthesis based on co-optimization and reward engineering. We assign human action penalty to be $c_{\text{p}} = 3, 3.5, 4$, and find that larger penalties lead to the human agent applying lower magnitude actions. To compensate for the penalty, the human agent tends to move joints earlier in the kinematic chain of the arm, such as the shoulder instead of the elbow, which can induce larger motions. As a result, the varying penalties lead to qualitatively different solutions for the co-optimization. Secondly, we vary the human agent by simulating different itch positions on their arm, where the itch position is unknown to the robot. We train co-optimized human and robot policies conditioned on different itch spots, which leads to qualitatively different actions for the human and the robot. With these two different axes, we can generate a population of human with different itch positions as well as activity levels. We are only using this as a first step to understand how different methods generalize, since we never expect to be able to capture

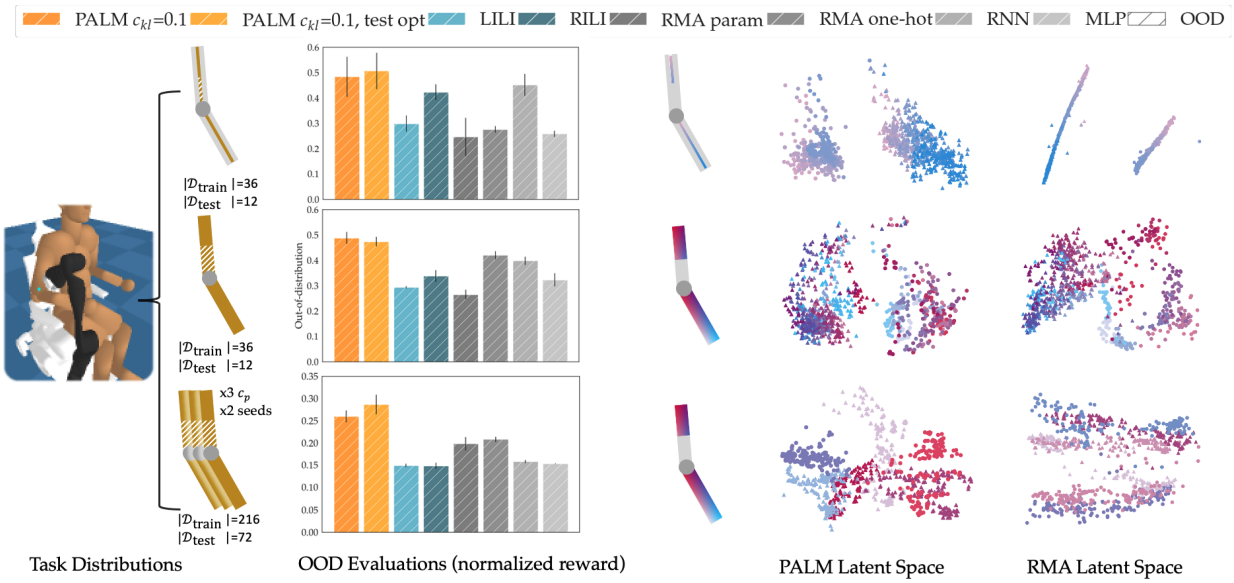


Figure 5.8: In assistive itch scratching, we sample humans by different itch positions and activity level (varying action penalty c_{p}). We visualize the in- and out-of-distribution humans on the two-link arm figures. We also visualize the embedding space of PALM and RMA, where we color-code the embeddings of in-distribution humans. Here we leave the embeddings of other baselines to supplementary material.

the diversity in humans perfectly. Indeed, figuring out a good synthetic population that can effectively model real-world humans is an important open question.

Extrapolating beyond training distribution As shown in Fig. 5.8, we divide the surface area on the human's arms into four equal portions from shoulder to wrist. We simulate 12 in-distribution humans from each of the three training training portions, as highlighted in brown in Fig. 5.8, and use 12 humans from the held-out section for out-of-distribution test. In total, we have 36 in-distribution humans and 12 out-of-distribution humans.

We construct three distributions in increasing order of difficulties. In the first distribution Γ_1, we sample itch positions from a line-shaped area on the outer side of the arm. In the second Γ_2, we sample itch positions from all the surface area around the arm. Note that Γ_1 and Γ_2 are constructed by setting $c_{\text{p}} = 3$. In the third Γ_3, we randomly sample from humans of $c_{\text{p}} = 3, 3.5, 4$, each trained with two different random seeds. This add in the extra complexity of human activity levels.

Human parameters at training time. Some baselines such as RMA require the training humans to be parameterized in some way to distinguish the different humans. In principle, one could always use one-hot IDs, but the quality of the learnt encodings and final performance of the method depends heavily on the quality of the parameterization. We

evaluate the RMA baseline both on a 36-dimensional one-hot vector (concatenated with the robot's observations) and also the task-specific parameterization we could come up with that characterizes the axes of variation. We use a three dimensional vector that includes x, y position of the itch position and the arm index. The third dimension indicates the action penalty.

We find PALM achieves better generalization result on the held-out humans across all three types of distributions. Interestingly, both RMA methods have access to more information during training than PALM, yet still perform worse. We also see that despite not knowing about the training distribution a-priori, PALM is able to learn the underlying structure in the human distribution in an unsupervised fashion purely from interaction.

Results. We follow a similar procedure as the reach environment to train itch scratching policy, and train for 240 epochs with 192 trajectories for batch size. As shown in Fig. 5.8, we observe PALM with action prediction has better generalization performance than other baselines. We see that in the simplified distribution $\mathscr{D}^1$, RILI and MLP have the best generalization performance among baselines, yet as the complexity of the training human distribution increases, they deteriorate.

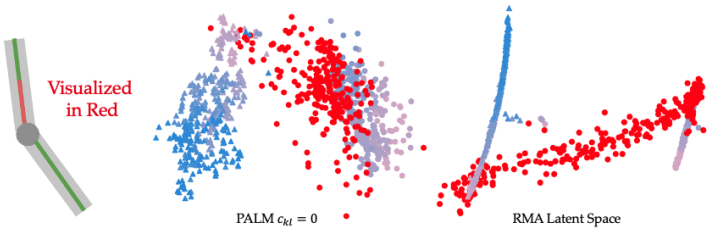


Figure 5.9: We visualize the embeddings of the OOD humans in assistive gym latent space. We use the second distribution, where all humans are sampled from a line-shaped region.

Visualizing the Latent Space To further investigate why PALM generalize better to OOD human than RMA baselines, We visualize the latent space of the "straight-line" distribution. As we see in Fig. 5.9, PALM can capture the structure in human training distribution as two clusters, and also correctly embed the OOD humans distribution as a part of the upper arm distribution. RMA-based methods, on the other hand, can discover the structure of training humans. Yet qualitatively, they fail the correctly embed the OOD humans in proximity to the upper arm distribution.

User Study

We conduct a user study with 9 users from age 22 to 29. We simplify the VR environment such that the user has four degrees of freedom: three on the shoulder and one on the elbow. We do not explicitly track the elbow and use the controller rotation and orientation to track the arm motions. We then convert the human pose deltas into actions in the environment. Note that the real user can exceed the action limit in assistive gym (`env.action_space_human.high`). This can be challenging for the robot because it has

Figure 5.10: The user study setup. Left and center are the user interacting with virtual robots through the HTC VIVE headset and the hand controller. The right is the first-person view in VR[46].

not seen such action magnitude during training. We use a red marker to remind the users if they have exceeded the action limit.

For each user, we test with two itch spots, one on the upper arm and one on the lower arm. We use randomized orderings of different methods. Each interaction is 150 simulation time steps and roughly six seconds of wall clock time. Within each method, we let the user interact for a total of five trials. We also find that the robot can be initialized in poses that collide with the human at that moment. We discard trials when such collisions are detected.

For training the robot controller, we synthesize humans of six different activity levels ($c_p = 0, 3, 10, 30, 60, 100$), each with two random seeds to simulate users of different activity levels. We then sample 16 itch spots from each human, which results in a total of 192 training humans. The training itch spots do not include the two spots that we test on.

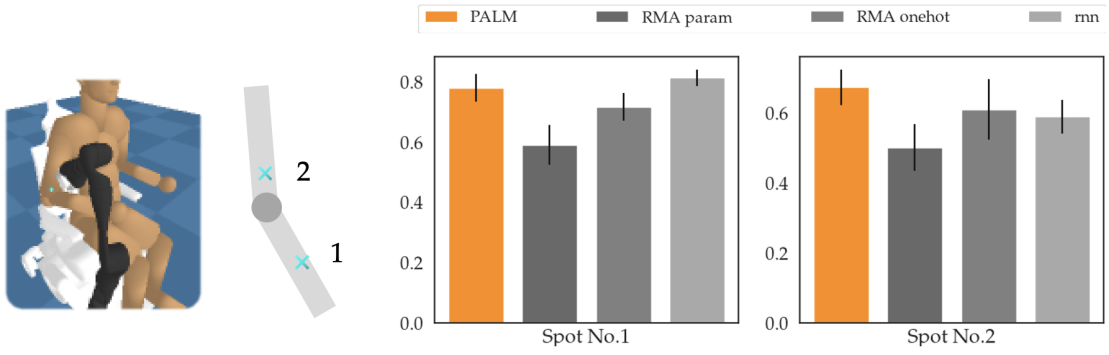


Figure 5.11: Left: visualization of the two itch spots in user study. Right: relative performance of different methods. The error bar indicates standard error.

For evaluation, we compute *success count*: the total number of time steps where the robot's tool is within 10 centimeters of the itch location. To account for different user

dexterity levels, we then normalize the result against the highest and the lowest *success count* on that itch spot for that user. We show the results in Fig. 5.11. Note that PALM yields high performance in both itch spots and outperforms the RMA baselines. We also find the performance of the RNN baseline to be comparable with PALM for the first task. We hypothesize that this might be due to the real users being able to quickly adapt to the robot's behaviour. This causes the performance of all the methods to be more similar than shown in the simulation experiments.

Comparison with other VAE baselines for sequential data

Note that our method relies on embedding human trajectories as sequential data into a latent space. Our implementation uses the final hidden state of RNN as the input to variational autoencoder. This is based on [16], which has been shown to be effective in embedding and generating sentences. Given that there are other different generative models for sequential data, our framework can be easily combined with them. In fact, wee believe coming up with a better model for human embedding is a future direction.

We hereby provide comparison with another different generative sequential model [34]. Different from [16] that uses only the final hidden state, they construct a latent space for every intermediate step in the sequential model. We keep all the experiment hyper-parameters the same, and concatenate the final hidden state with observation as input to the robot policy. We show the results in

Normalized Reward	Distribution	Our Method [16]	Baseline [34]	RNN
Assistive Reacher	IND	**0.72 ± 0.02**	0.66 ± 0.02	0.62 ± 0.02
	OOD	**0.38 ± 0.10**	0.11 ± 0.01	0.27 ± 0.09
Itch Scratching $\mathscr{D}^1$	IND	0.75 ± 0.01	0.45 ± 0.04	**0.79 ± 0.01**
	OOD	**0.48 ± 0.08**	0.32 ± 0.01	0.25 ± 0.08
Itch Scratching $\mathscr{D}^2$	IND	0.58 ± 0.03	**0.70 ± 0.03**	0.44 ± 0.01
	OOD	**0.49 ±.0.02**	0.31 ± 0.05	0.40 ± 0.02
Itch Scratching $\mathscr{D}^3$	IND	**0.34 ± 0.02**	0.23 ± 0.22	0.18 ± 0.01
	OOD	**0.25 ± 0.01**	0.13 ± 0.01	0.16 ± 0.01

Table 5.2: Comparison with RNN-VAE baseline [34]

Additional Experiment: Bed Bathing Task

We further evaluate the performance of PALM in another assistive robotics task: robot-assisted bathing. This task is a modified version of the bathing task introduced in Assistive Gym [47]. In this task, we have a human lying on a tilted bed, with a robot mounted on the nightstand. The human can move their right arm, and there is an identified patch

of skin to be cleaned. Unlike the original bed bathing task, where the entire right arm is covered in target points to be cleaned, we instead initialize a fixed size region of points to be cleaned at a uniform randomly selected location along the surface of the right forearm. The patch spans 10 centimeter along and 150 degrees around the forearm. Only the human knows the center position of the patch, and the robot must infer the location of the patch based on observations of the human motion. The points along the body are cleaned whenever the robot initiates contact with the spot using its end-effector and applies a positive normal force. The task reward is based on how many points are cleaned.

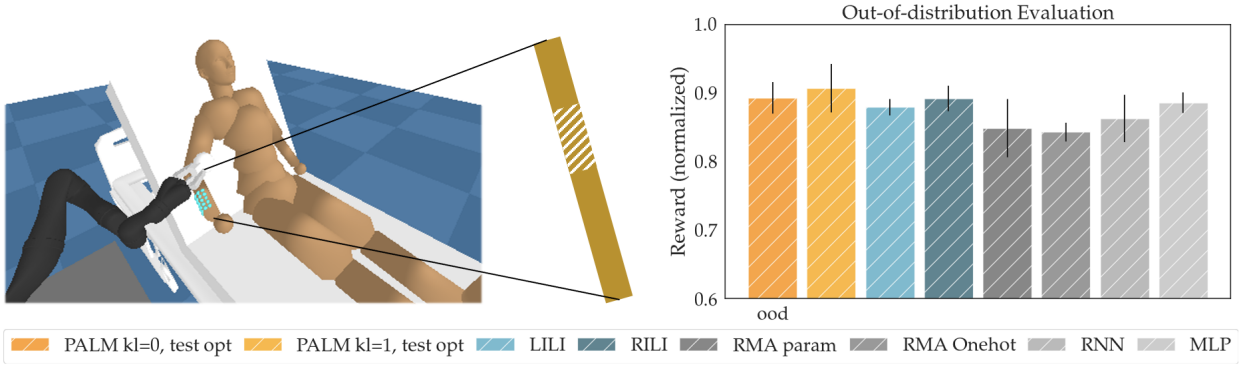


Figure 5.12: Visualization of the bed bathing task (left), where the human lies on a tilted bed with a table-mounted robot to clean an area on their arm. The area is random initialized (middle) on the forearm, where we hold out a quarter of the length as the held out distribution. The results of the held-out distribution is visualized on the right.

We generate synthetic humans by co-optimizing humans and robots conditioned on the centroid position of the region to be cleaned. During co-optimization, we adopt the formulation in [47] where both the human and the robot observe the centroid position along the human forearm. This helps induce collaborative behaviour for the human and robot policies, where we then use the resulting human as the synthetic population. During training, we blind of robot policy of the centroid position. We use the co-optimized robot (observes the centroid position) as the oracle for querying expert actions for Behaviour Cloning. We sample 18 humans from the training distribution, and save 6 humans from the held-out distribution as the out-of-distribution evaluation. We use the same hyperparameters as the itch scratching experiment.

As we see in the results in Fig. 5.12, PALM achieves better out-of-distribution results compared to the baseline methods. We also observe that this performance gap is smaller than the itch scratching task. We believe this is due to the nature of the bed bathing task, where a robot controller that maintains contact with the human's forearm can be sufficient for solving the task if the human policy learns to move and rotate their forearm accordingly to help the robot.

Limitations and Failure Cases

Although PALM achieves good average-case performance, it works best with humans sampled near the training distribution. If we pair the robot with an adversarial human, PALM is likely to fail as it lacks a fall-back safety policy.

The major limitation of PALM is the requirement of generating a human population. While we provide one way to generate human populations based on weighted human-robot co-optimization, we lack ways to systematically generate diverse and realistic human motions. One important direction for future work is to incorporate real user data to create training populations. Improving the realism of the training human population is likely a crucial step to supporting transfer to real partners.

One future direction is extending to settings with one patient and one human caregiver. While our framework still applies, this leads to new challenges including (1) learning a joint or separate latent space for human patient and caregiver, (2) modeling a population of human caregivers for training in simulation, and (3) modeling communication between the human caregiver and patient.

5.5 Conclusion

Generalization is an important task for assistive robotics, and in this paper, we formulate a problem setting that focuses on Out-Of-Distribution users. To that end, we contribute a framework PALM for learning a robot policy that can quickly adapt to new partners at test time. PALM assumes a distribution of training humans and constructs an embedding space for them by learning to predict partner actions. We can further adapt this embedding at test time for new partners. Experiments show that PALM outperforms state-of-the-art approaches. We are excited by the potential of using PALM to enable robotic assistance in the future.

Chapter 6

Robustness via Natural-Adversarial Frontier

This chapter explores how we can actively synthesize adversarial human agents for failures, and use this adversarialness to characterize the safety of Human AI systems.

6.1 What is Adversarial Synthesis for Human AI Systems?

Deploying assistive robots in healthcare facilities and homes hinges crucially on their robustness, reliability, and the assurance that they can safely interact with individuals of all ages, from children to older adults. The worst-case scenario would involve an errant robotic arm causing injuries. However, slight variations or unpredictable elements in the operational environment can influence the behavior of learned robotic policies. Consequently, we must thoroughly examine the robot's response not just to regular situations but also to potential scenarios where unexpected events may unfold.

The complexities of stress testing robot policies, however, can be exceptionally high. For instance, verifying the robustness of a dish-loading robot in all kitchen configurations is highly costly [181]. Verifying human-robot interaction applications, in comparison, is even more challenging due to the fact that the human partner performs independent actions, changing the state that the robot's policy responds to, and potentially inducing distribution shift.

The common approach to probing the policy's robustness — perturbing inputs x to the model — would translate here to perturbing the human partner's actions. But this can paint a misleading picture of the robustness of an assistive policy: while in theory any human action is possible, in practice humans won't move in arbitrary ways. A robot policy might thus fail under small *adversarial* perturbations but work under quite large *natural* deviations in human motion.

We thus propose to measure robustness to *natural* variations in human motion. This begs the question of how to define "natural". We propose a generative approach: given

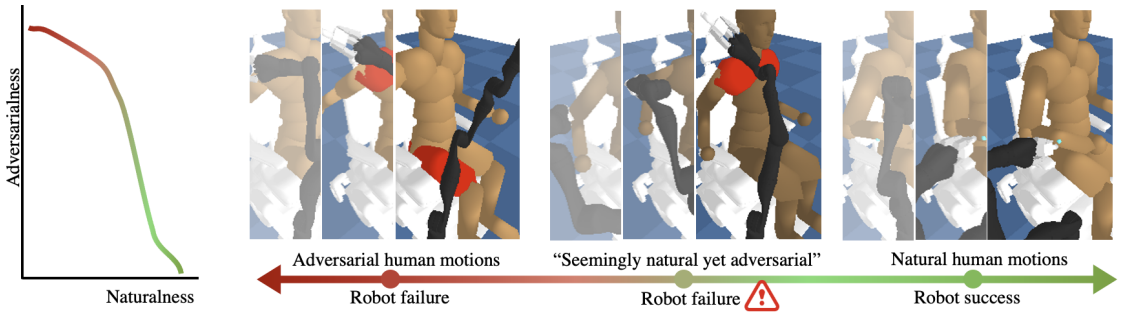


Figure 6.1: We propose measuring assistive robustness by considering the *naturalness* of human motions, and analyzing the entire natural-adversarial frontier: points with natural human motion that lead to good robot performance (left), unnatural motions that easily break the policy (right), and natural motions that still lead to failures (center). The full frontier paints a more useful picture of robustness than looking at a single point, and can be reduced to a scalar using the area under the curve (AUC): lower AUC is indicative of higher robustness.

a dataset of motions — demonstrated by the human in the target task or sampled from a trusted human model/simulator [30, 99] — we train a GAN [55, 71] and definite naturalness via the learned discriminator.

Next comes the question of *how* unnatural we allow human motions to be in our search for adversarial attacks against the robot: it is easy to break most policies if we allow motions that are very unnatural, and we might not find the interesting failure cases if we restrict the motion to be too close to what was seen in training. We thus advocate that robustness should not be assessed by looking at a single threshold, but rather by looking *at the entire natural-adversarial frontier*.

> *Our proposal is to measure assistive robustness by considering the naturalness of human motions and analyzing the entire natural-adversarial frontier.*

This Pareto frontier likely contains points of poor robot performance caused by unnatural human motions and points of good robot performance interacting with naturally moving humans. What is interesting lies in between: there exist natural motions that can lead to unexpected failures as in Fig. 6.1, leading to a high area under the curve.

We introduce RIGID, a method that constructs the natural-adversarial frontier by training adversarial human policies that trade-off between minimizing robot reward and acting in a human-like way, as measured by the discriminator [1]. We construct the frontiers for different policies — one trained to naively collaborate with the human, and others trained with robust Reinforcement Learning methods (e.g. by using populations) [68,

[1]While in our work we use a simulated human to generate data for the discriminator, in general, this data can also come from human-human or human-robot interaction in the task.

40]. While we find that prior robust RL methods improve policy robustness measured by RIGID, we are able to uncover edge cases where natural motions (natural with regard to our simulated human) cause robot failures. We then use RIGID to improve robustness by fine-tuning regular RL using data points identified by RIGID. Finally, we conduct a user study with naive users and an expert who attempts to lower the robot's reward. We find that RIGID can generate failure cases more effectively than manual effort, and is predictive of deployment performance.

6.2 Natural-Adversarial Robustness

In this section, we first introduce the robotic assistance problem in Fig. 6.2, where the goal is to assist a human in a partially observable setting — the human may have hidden information, such as intent or preferences, that needs to be inferred on the fly. We then formulate the problem of adversarial but natural human motion, and reduce it to the optimization of an objective trading off between the two.

Assistance as a Two-Player Dec-POMDP. Following prior work [63, 179, 68] we model an assistive task as a two-agent, finite horizon decentralized partially-observable Markov decision process (Dec-POMDP), defined by the tuple $\langle S, \alpha, A_R, A_H, \mathcal{T}, \Omega_R, \Omega_H, O, R \rangle$. Here S is the state space and A_R and A_H are the human's and the robot's action spaces, respectively. The human and the robot share a real-valued reward function $R : S \times A_R \times A_H \rightarrow \mathbb{R}$; however, we assume that the reward function is not fully observed by the robot, i.e., some of the reward function parameters (e.g., the specific goal location or objective of an assistive task) are in the hidden part of the state. $\mathcal{T}$ is the transition function where $\mathcal{T}(s' \mid s, a_R, a_H)$ is the probability of transitioning from state s to state s' given $a_R \in A_R$ and $a_H \in A_H$, Ω_R and Ω_H are the sets of observations for the robot and human, respectively, and $O : S \times A_R \times A_H \rightarrow \Omega_R \times \Omega_H$ represents the observation probabilities. We denote the horizon of the task by T.

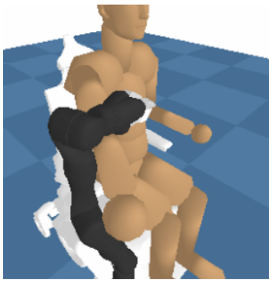


Figure 6.2: Assistive task: itch scratching with unknown itch locations. See Sec. 6.6 for more details.

Dataset Aggregation (DAgger) [150] is a framework for passively incorporating failure cases into robot learning. One limitation of DAgger is that it requires online interaction with real humans in order to improve robustness – failing, and having the human correct or otherwise augment the data and retraining. This is undesirable in assistance and may cause harm to users. In contrast, our framework is an instance of active learning [67] that looks for failure cases of the robot policy before deployment. This way, we actively discover failure cases in simulation without real-world consequences.

Generative adversarial imitation learning (GAIL) [71] is a promising method for producing motions that mimic a set of demonstrations by training a discriminator on a dataset to provide the reward for a reinforcement learning agent. Researchers have exper-

imented with variants of GAIL and found that it leads to high-quality motions in graphics [139], locomotion [48], and even controllable natural motions [138]. In this work, we build on the existing techniques for training GAIL to generate human motions that are perceived as natural. This allows us to study assistive robustness: by optimizing for the joint objective of being both natural and adversarial, we search for a range of plausible and typical human motions that create catastrophic failures for robot policies in assistive settings.

Adversarial Human Motions. Let a human trajectory denote a sequence of states and actions $\tau = (s_1, a_1^H, ..., s_T, a_T^H)$. and let function π_H be the human policy that maps from local histories of observations $\mathbf{o}_t^H = (o_1^H, \ldots, o_t^H)$ over Ω_H to a probability distribution over actions. We define for an arbitrary human policy π_H, its corresponding assistive robot policy $\pi_{R(H)}$, which is the robot policy that is optimized to collaborate best with π_H with respect to the joint reward R: $\pi_{R(H)} = \arg\max_{\pi_r}[\sum_t R(\pi_R, \pi_H)]$.[2]

We can then define an adversarial human policy $\tilde{\pi}_H$ with respect to an assistive robot policy. The policy $\tilde{\pi}_H$ minimizes the overall performance under the constraint that $\tilde{\pi}_H$ is similar to the original policy π_H[3] as measured by an f-divergence measure $D_f(\tilde{\pi}_H || \pi_H) \leq \delta$, and adjustable coefficient δ controls the allowable perturbation set in which the adversarial human policy $\tilde{\pi}_H$ deviates from π_H, similar to adversarial perturbations in computer vision [37]:

$$\tilde{\pi}_H(\pi_{R(H)}, d) = \arg\min_{\pi'_H}[R(\pi_{R(H)}, \pi'_H)] \quad \text{s.t. } D_f(\pi'_H || \pi_H) \leq \delta \tag{6.1}$$

$\tilde{\pi}_H$ and π_H are typically modeled by neural networks, making it difficult to directly compute their differences. We hereby introduce methods to compute their f-divergence via resulting trajectories.

Approximating the Divergence Measures such as KL divergence [161, 76, 99, 183], χ^2 [116] divergence are commonly used to characterize the distance between two probability distributions. In the context of policy learning, this can be achieved by training a discriminator $\mathcal{D} : \tau \to \mathbb{R}$ to distinguish between trajectories sampled from $\tilde{\pi}_H$ and π_H. More specifically, $\mathcal{D}$ assigns a score $\mathcal{D}(\approx) := \mathbb{E}_{\tau\sim\pi}[\mathcal{D}(\tau)]$ to any policy π. It is trained to assign low scores to trajectories drawn from the true human policy π_H and high scores to trajectories drawn from the adversarial policy $\tilde{\pi}_H$. We choose the LS-GAN objective [116] for training $\mathcal{D}$:

$$\mathcal{D} = \arg\min_{\mathcal{D}} \ \mathbb{E}_{\tau\sim\tilde{\pi}_H}\left[(\mathcal{D}(\tau) - 1)^2\right] + \mathbb{E}_{\tau\sim\pi_H}\left[(\exp\{\mathcal{D}(\tau) + 1)^2\right] \tag{6.2}$$

Previous work [116] has proven that this represents χ^2 divergence when trained to optimality: $D_{\chi^2}(\tilde{\pi}_H || \pi_H) = \mathbb{E}_{\tau\sim\tilde{\pi}_H}[\mathcal{D}(\tau)^2]$. We illustrate this in Sec. 6.6. Note that while we

[2]For the sake of simplicity, we use a modified R to denote the expected reward of the rollouts of human-robot policy pairs as $R(\pi_R, \pi_H) := \mathbb{E}_{a_r^t\sim\pi_r(s^t)}\mathbb{E}_{a_h^t\sim\pi_H(s^t)}\mathbb{E}_{s_1}[\sum_t R(s^t, a_r^t, a_h^t)]$.

[3]Typically, robot policies are trained via RL as a best response to a human model [30]; offline RL [72] is also possible and bypasses a human model, in which case the discriminator can be trained directly on the offline data.

focous on χ^2 divergence in this paper, in practice one may use other divergence measures such as KL divergence and such as MMD (Maximum Mean Discrepancy) [58].

6.3 Different Choices of Naturalness Measures

As discussed in Sec. 6.2, we would like to constrain the adversarial policy $\tilde{\pi}_H$ to be similar to the original π_H with respect to some f-divergence metric of the policy distribution. Commonly used divergence function include χ^2 divergence, KL divergence, Wasserstein distance, etc. Such divergence measures are difficult to be estimated and optimized in Eq. (6.9). Thus, we present the variational form of them. The idea is that we can represent them using a discriminator optimized with a specially designed loss function. The advantage is that the discriminator is differentiable, compatible with Eq. (6.9), and is provably equivalent to the corresponding divergence measures when trained to optimality.

Proof on Discriminator and χ^2 Divergence

Based on [116], we train the discriminator $\mathcal{D}(\ll)$ and the policy $\tilde{\pi}(\tau)$ using the following loss.

$$\mathcal{D} = \arg\min_{\mathcal{D}} \frac{1}{2}\mathbb{E}_{\tau\sim\pi_H}\big[(\mathcal{D}(\tau)-b)^2\big] + \frac{1}{2}\mathbb{E}_{\tau\sim\tilde{\pi}_H}\big[(\exp\{\mathcal{D}(\tau)-a)^2\big] \tag{6.3}$$

$$\tilde{\pi}_H = \arg\min_{\pi} \frac{1}{2}\mathbb{E}_{\tau\sim\pi_H}\big[(\mathcal{D}(\tau)-c)^2\big] + \frac{1}{2}\mathbb{E}_{\tau\sim\pi}\big[(\exp\{\mathcal{D}(\tau)-c)^2\big] \tag{6.4}$$

Here a, b, c are constants, π_H is the canonical data distribution of the interaction, learned from human data or designed a-priori. Note that even though we work in the policy learning settings, $\mathcal{D}$ and $\tilde{\pi}_H$ are analogous to the discriminator and the generator in the GAN literature. $\mathcal{D}$ and $\tilde{\pi}_H$ are optimized iteratively till convergence. We hereby prove that the resulting $\tilde{\pi}_H$ minimizes the Pearson χ^2 divergence. We first derive the optimal discriminator for a fixed π in Eq. (6.3) as:

$$\mathcal{D}^*(\tau) = \frac{b\pi_H(\tau) + a\pi(\tau)}{\pi_H(\tau) + \pi(\tau)}$$

We can then formulate the objective for $\tilde{\pi}$ in Eq. (6.4) as:

$$\begin{aligned}
2\text{Loss}(\pi) &= \mathbb{E}_{\tau\sim\pi_H}\big[(\mathcal{D}(\tau)-c)^2\big] + \mathbb{E}_{\tau\sim\pi}\big[(\exp\{\mathcal{D}(\tau)-c)^2\big] \\
&= \mathbb{E}_{\tau\sim\pi_H}\big[(\frac{b\pi_H(\tau)+a\pi(\tau)}{\pi_H(\tau)+\pi(\tau)}-c)^2\big] + \mathbb{E}_{\tau\sim\pi}\big[(\exp\{\frac{b\pi_H(\tau)+a\pi(\tau)}{\pi_H(\tau)+\pi(\tau)}-c)^2\big] \\
&= \int_\tau \frac{((b-c)\pi_H(\tau)+(a-c)\pi(\tau))^2}{\pi_H(\tau)+\pi(\tau)}\mathrm{d}\tau, \quad \text{set } b-c=1 \text{ and } b-a=2 \\
&= \int_\tau \frac{(2\pi(\tau)-(\pi(\tau)+\pi_H(\tau)))^2}{\pi_H(\tau)+\pi(\tau)}\mathrm{d}\tau \\
&= \chi^2_{\text{Pearson}}(\pi_H+\pi||2\pi)
\end{aligned}$$

This means that the optimal $\tilde{\pi}_H$ optimizes for the χ^2 divergence. In practice, we select $b=1, a=-1, c=0$ so that

$$\mathcal{D} = \arg\min_{\mathcal{D}} \ \mathbb{E}_{\tau\sim\tilde{\pi}_H}\big[(\mathcal{D}(\tau)-1)^2\big] + \mathbb{E}_{\tau\sim\pi_H}\big[(\exp\{\mathcal{D}(\tau)+1)^2\big] \tag{6.5}$$

$$\tilde{\pi}_H = \arg\min_{\pi} \frac{1}{2}\big[\mathcal{D}(\tau)^2\big] \tag{6.6}$$

This is the formulation we used in Eq. (6.2). Q.E.D.

Proof on Discriminator and KL Divergence

Based on [76, 99], we use the following objective for the discriminator $\mathcal{D}(\tau)$ and the policy $\tilde{\pi}(\tau)$:

$$\mathcal{D} = \arg\min_{\mathcal{D}} \ \mathbb{E}_{\tau\sim\tilde{\pi}_H}\big[\log(1+\exp\{-\mathcal{D}(\tau)\})\big] + \mathbb{E}_{\tau\sim\pi_H}\big[\log(1+\exp\{\mathcal{D}(\tau)\})\big] \tag{6.7}$$

$$\tilde{\pi}(\tau) = \arg\min_{\pi} \mathbb{E}_{\tau\sim\pi}\big[\mathcal{D}(\tau)\big] \tag{6.8}$$

We can rewrite Eq. (6.7) as:

$$\mathcal{D} = \arg\min_{\mathcal{D}} \int \log\big(1+\exp\{-\mathcal{D}(\tau)\}\big)\pi(\tau) + \log\big(1+\exp\{\mathcal{D}(\tau)\}\big)\pi_H(\tau)\mathrm{d}\tau$$

The integral is minimized if and only if the integrand is minimized for all τ, that is

$$\forall\tau, \mathcal{D} = \arg\min_{\mathcal{D}} \log\big(1+\exp\{-\mathcal{D}(\tau)\}\big)\pi(\tau) + \log\big(1+\exp\{\mathcal{D}(\tau)\}\big)\pi_H(\tau)$$

We can then show that the value for $\mathcal{D}(\tau)$ is $\mathcal{D}(\tau) = \log\big(\frac{\pi(\tau)}{\pi_H(\tau)}\big)$. Plugging this into the objective function in Eq. (6.8), we get

$$\text{Loss}(\pi) = D_{\text{KL}}(\pi(\tau)||\pi_H(\tau))$$

This means that the optimal $\tilde{\pi}_H$ optimizes for the KL divergence. Q.E.D.

Discriminator-free Method: Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) [58] is a kernel-based statistic test used to measure the difference between two distributions, and can be used as a loss/cost function in machine learning algorithms for density estimation.

Formally, given random variables X, Y, a feature map ϕ mapping X to another feature space $\mathcal{F}$ such that $\phi(X) \in \mathcal{F}$, we can use the kernel trick to compute the inner product of X, Y in $\mathcal{F}$ as $k(X, Y) = \langle \phi(X), \phi(Y) \rangle_{\mathcal{F}}$. We define feature means as a probability measure P on X, which takes $\phi(X)$ and maps it to the means of every coordinate of $\phi(X)$:

$$\mu_P(\phi(X)) = \big[[\phi(X)_1], ..., [\phi(X)_m]\big]^T$$

The inner product of feature means of $X \sim P$ and $Y \sim Q$ can be written as

$\langle \mu_P(\phi(X)), \mu_Q(\phi(Y)) \rangle = \mathbb{E}_{P,Q}[\langle \phi(X), \phi(Y) \rangle_{\mathcal{F}}]$, Maximum Mean Discrepancy is defined as

$$\text{MMD}^2(P, Q) = ||\mu_P - \mu_Q||^2_{\mathcal{F}}$$

In practice, we can leverage MMD to measure the distance between $\tilde{\pi}_H$ and π_H, by collecting datasets of trajectories from each policy, and computing the MMD distance of the two sets. We can then plug in MMD to replace D_f in Eq. (6.10). This way, the divergence can be directly estimated without training a dissec:3-problem-definitioncriminator. We use the MMD implementation in [189], which uses a radial basis function as the kernel function.

Adversarial Frontier While previous work in Computer Vision (CV) [27, 37] assign the perturbation set to a fixed value, it is unclear what value we should assign to δ in the context of Human-Robot Interactions. Instead, we consider scanning over all possible values of δ to find the adversarial human policy under all different naturalness criteria. In other words, we consider all levels of "naturalness" in human motions. We consider the Lagrange dual function of the constrained optimization in Eq. (6.1):

$$L(\pi_R, \lambda) = \max_{\pi'_H} \big[- R(\pi_R, \pi'_H) - \lambda \cdot D_f(\pi'_H || \pi_H) + \lambda \cdot \delta \big] \tag{6.9}$$

$$\tilde{\pi}_H(\pi_R, \lambda) = \arg\max_{\pi'_H} \big[- R(\pi_R, \pi'_H) - \lambda \cdot D_f(\pi'_H || \pi_H) \big] \tag{6.10}$$

We have reduced the constrained optimization in Eq. (6.1) to an unconstrained optimization in Eq. (6.10) that balances "adversarialness" (playing an adversary to the robot by optimizing for $-R(\pi_R, \cdot)$) and "naturalness" (staying close to the canonical interaction by minimizing a divergence metric $\mathcal{D}(\cdot)$). The parameter λ provides a knob that we can use to trade off how adversarial and how natural we would like $\tilde{\pi}_H$ to be. Setting $\lambda \to \infty$ results in a policy $\tilde{\pi}_H$ that closely resembles π_H and yields a high reward. On the other hand, setting $\lambda = 0$ leads to a policy $\tilde{\pi}_H$ that is purely adversarial and causes harm to the assistive task by inverting the environment reward. By selecting the naturalness parameter $\lambda \in [0, \infty)$, we arrive at a spectrum of human motions that interpolate between

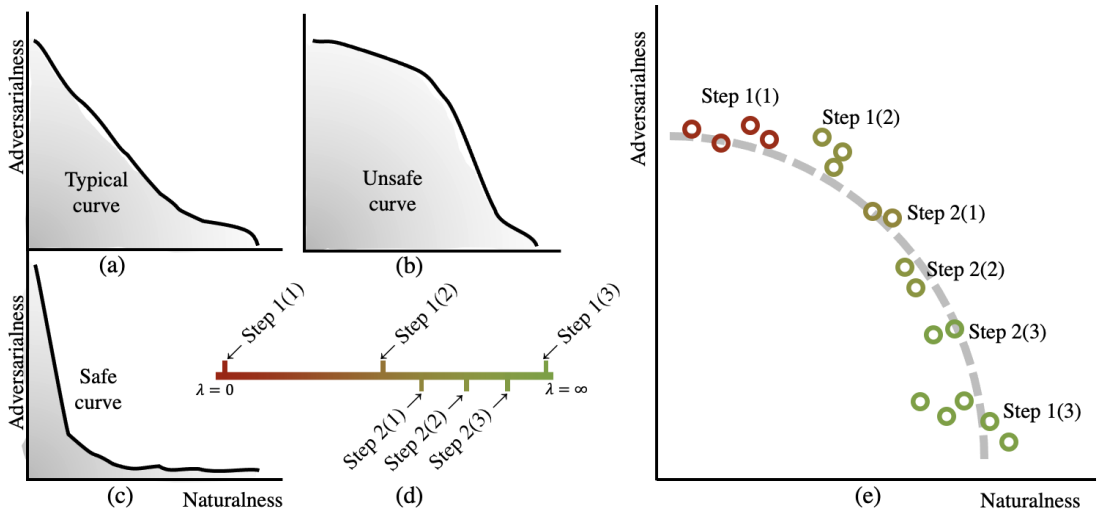


Figure 6.3: Left: different types of Natural-Adversarial curves. (b) has a higher AUC means it is less robust and more prone to unsafe behaviors. The ideal (c) curve looks like a sharp drop. Right: when running RIGID in Alg. 5, we first sample λ's evenly in the log space over $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ in (d). We then iteratively refine our picks by sampling more finely over selected intervals to gain better coverage of the curve in (e).

adversarial and natural. This is meaningful in Human-Robot Interaction because while we may not need to worry about purely adversarial human behaviors, as they are less likely to happen in reality, we need to take precaution against human behavior that appears natural, yet causes robot failures.

Given these definitions, we can in principle compute the naturalness and adversarialness of all possible human motions, and plot them in a 2D coordinate system. This motivates the Natural-Adversarial curve — the Pareto frontier of non-dominated policies. These are policies that, compared to any other ones, have either a better adversarial score, or a better naturalness score, or both.

This theoretical frontier enables us to describe robustness as the Area Under the Curve (AUC) — the more concave, the safer the policy is. We then propose a practical algorithm (see Sec. 6.4) called RIGID that uses generative adversarial imitation learning in a multi-objective optimization fashion to produce naturally adversarial human motions. We then utilize RIGID in a tree-search fashion to find the assistive robustness curve.

Robust Robot Policy Now that we have formally defined adversarial human policies, we define a robust robot policy as the optimal policy under all possible human adversarial policies $\tilde{\pi}_H$:

$$\begin{aligned} \pi_{R_{\text{robust}}}(\pi_H, \lambda) &= \arg\max_{\pi_R} \min_{\tilde{\pi}_H \in \Delta} [R(\pi_R, \tilde{\pi}_H)] \quad \text{where } \Delta = \{\tilde{\pi}_H \big| D(\tilde{\pi}_H || \pi_H) \leq \delta\} \\ &= \arg\max_{\pi_R} \Big\{ \max_{\tilde{\pi}_H} \big[-R(\pi_R, \tilde{\pi}_H) - \lambda \cdot \mathcal{D}(\tilde{\pi}_H) \big] \Big\} \end{aligned} \tag{6.11}$$

Finding such robust robot policies under different λ is a key goal of designing reliable robotic assistants. However, solving this nested optimization is intractable. Coming up with effective ways to approximate the solution remains an open question [40]. **TODO: not sure where this last sentence is going. Needs a better end to set up the following section.**

6.4 Computing the Natural-Adversarial Frontier

In this section, we propose RIGID — a practical method to obtain the Adversarial Frontier in Sec. 6.2. Having motivated the concept of an adversarial human policy $\tilde{\pi}_H(\pi_R, \lambda)$ and the naturalness parameter, λ, in Sec. 6.2, we now discuss the Natural-Adversarial Pareto frontier (Sec. 6.4). We then introduce an algorithm, RIGID (Sec. 6.4), that efficiently approximates the natural-adversarial curve.

Connecting The Dots: the Natural-Adversarial Curve

The Coordinate Axis: In Fig. 6.3, we plot a range of human motions, where the x-axis denotes the normalized numerical "naturalness" from 0 to 1. We implement this as the accuracy prediction from a discriminator $\mathcal{D}$ trained on a dataset of π_H, as described in Eq. (6.2). The y-axis denotes the normalized numerical adversarialness, defined as the inverse of environment reward R. This coordinate system allows us to plot all possible human motions that exist, as every human motion corresponds to an $(x, y) = (\text{naturalness}, \text{adversarialness})$ pair. Among all such human motions, there exist a frontier. Every solution to Eq. (6.10) for a specific λ corresponds to a point on the Pareto frontier. Note that in practice, we can cap the maximum adversarialness to a set value, as certain human motions can create arbitrarily low rewards: e.g., repeatedly hitting the robot.

Points Along the Frontier: To solve for Eq. (6.2), we start with sampled trajectories from π_H (in our experiments, a human model; in general, this can also be data collected from human-human or human-robot interaction). For a specific λ, we adapt GAIL [71] to our setting: we interleave training the adversarial policy $\tilde{\pi}_H$ and the discriminator $\mathcal{D}$ by iteratively (a) doing one step of policy optimization on Eq. (6.10) to adapt the policy and (b) doing one gradient step on Eq. (6.2) to adapt the discriminator. In practice, we use PPO [160, 92] as our policy optimization algorithm, and we use LS-GAN [116] with noise annealing to ensure the stability of the training process.

To plot the resulting $\tilde{\pi}_H$ in the natural-adversarial coordinate, we compute naturalness (x coordinate) by sampling trajectories $\tau \sim \tilde{\pi}_H$, and calculate the mean likelihood of $\mathcal{D}(\tau) < 0$. This is equivalent to having the discriminator classify the trajectories as from the original human model. We define adversarialness (y coordinate) as negative robot reward normalized to [0, 1].

To perform the RIGID algorithm to find the Natural-Adversarial Frontier Curve, we sweep over $\lambda \in [0.00001, 10]$. We perform RIGID algorithm in an iterative refinement manner as in Alg. 5. We keep 3 separate RIGID histories over 3 different random seeds. During each iteration, we select 6 new λ's for each seed. We terminate after three iterations. This results to a total of 54 RL runs per curve.

Plotting in the Natural-Adversarial Coordiate We set the naturalness (x value) of the resulting policies as the mean prediction result from the discriminator. To compute the adversarialness, we normalize the negative robot reward in [200, 1400] range, and clip values that exceed this range to the boundary. We find this range of manually performing different motions VR to find the mean negative reward values of natural motions as well as adversarial human behaviors that lead to failures.

Algorithm 5 RIGID

Maximum refinements d, samples per round k
Upper and lower bounds on λ: λ_{max}, λ_{min}
procedure RIGID(λ_{min}, λ_{max}, d)
  $\Lambda_{\text{all}} = []$, $S_{\text{nat}} = []$, $S_{\text{adv}} = []$
  $\lambda'_{\text{min}} = \lambda_{\text{min}}$, $\lambda'_{\text{max}} = \lambda_{\text{max}}$
  for $i = 1, \ldots, d$ **do**
    Find k evenly spaced $e_1 ... e_k$ in $[\log \lambda'_{\text{min}}, \log \lambda'_{\text{max}}]$
    for $j = 1, \ldots, k$ **do**
      $\lambda_j = \exp(e_j)$
      Λ_{all}.append(λ_j)
      Compute $\tilde{\pi}_H$ using Eq. (6.10) and λ_j
      Compute nat_j and adv_j for $\tilde{\pi}_H(\lambda_j)$
      S_{nat}.append(nat_j), S_{adv}.append(adv_j)
    end for
    λ'_{min}, λ'_{max} = LARGEST-JUMP(S_{nat},Λ_{all})
  end for
  return S_{nat}, S_{adv}
end procedure

Algorithm

Below we provide the pseudo-code for the LARGST-JUMP procedure. Give a list of noisily increasing values — in our case, a list of noisily increasing "naturalness" values as we increase λ, the goal of LARGST-JUMP is to identify the two values, adjacent or not, that have the largest gap between them. This enables iterative refinement in Alg. 5.

Approximating the Curve: Once we sample a sufficient number of λs (see Sec. 6.4 on how to do this) from $[0, \infty)$, we connect the outermost points. The resulting curve characterizes the frontier of adversarial human motions under different levels of naturalness

constraints (Eq. (6.10)) — or different perturbation sets (Eq. (6.1)). Intuitively, under the same naturalness value, lower adversarialness corresponds to more robust robot policies. Fig. 6.3 shows three different hypothetical curves, visualizing how the shape of the curve connects to the robustness of the policy.

The AUC Score as a Scalar Robustness Metric: Given a Natural-Adversarial curve, we compute the Area Under Curve (AUC) to quantitatively measure robustness as a single scalar. Lower AUC generally means stronger robustness, although there may be outlier human policies that can trigger unsafe behaviors, and a full plot of the curve is still necessary.

Sampling Useful Trade-Offs

When we gradually increase λ, the resulting adversarial human motion often goes through mode changes, which lead to sudden increases in naturalness. Motivated by this, we need to select $\lambda \in [0, \infty)$ in a non-uniform manner, by procedurally expanding between the two λ's where naturalness increases the most. We do this through an iterative refinement approach by finding the largest gap in naturalness and increasing the sampling density there.

Our algorithm's pseudocode is shown in Alg. 5. The lists S_{nat} and S_{adv} keep track of the corresponding naturalness and adversarialness values of points along the Pareto frontier. We initialize $\lambda_{\min} > 0$ to be the smallest λ we consider, and $\lambda_{\max}$ a sufficiently large number. We denote all the λ considered so far as Λ_{all}. The procedure LARGEST-JUMP looks at all λ selected so far and selects two λ values between which the naturalness changed the most. See Sec. 6.4 for pseudo-code for the algorithm.

Improving GAN training

While GANs are knownly difficult to train, there exists a large number of practical tricks in improving GAN training. We find that the most helpful tricks are: adding noise with annealing. LS-GAN, gradient penalty, and applying different weights for expert and human data. We experimented with training discriminators on concatenated observation and action, and find that it does not lead to much change. Thus we end up using observation-only discriminators.

Because human joints move at different rates and scales, it is important to add different amounts of noise to different joint observations. We compute the joint movements from the existing dataset [46], and multiply each joint movement's standard deviation by a factor of 10. We then anneal this by a decay rate of 0.98. We apply a gradient penalty of 0.3. We also set the expert loss rate in GAN as 4, and the agent loss rate as 1. This helps prevent the discriminator from overfitting to the agent data distribution and collapsing early in the training.

Computaional Complexity

To generate the adversarial curve in Sec. 6.5, we perform three scans over λ space, each time launching 18 different RL training in parallel for human policy learning. The three scans can be parallelized, and in total we perform 54 policy learning. Within each learning process, we run the PPO algorithm for 120 iterations, 4800 steps per iteration. This results in total of 0.5M environment timesteps. On Intel Xeon Skylake 6130 @ 2.1 GHz CPU, each policy learning 12 CPU core hours. In total, each Natural-Adversarial uses 648 CPU core hours.

Data Requirement

To train the LS-GAN as Naturalness Measure, we collect 40 canonical trajectories, each with 100 timesteps, which corresponds to 15 seconds of human-robot interactions. In total, we require 10 minutes of canonical demonstrations.

How to Compute Natural-Adversarial Curve on a New Domain?

In order to compute the Natural-Adversarial Curve on a new domain, we suggest the following steps:

1. Determine the key features. In the itch-scratching task, we find that robot poses, velocities and contact forces are the key features that determine whether the trajectories are dangerous. The naturalness measure is defined by these key features.

2. Determine the size of the canonical dataset. One may steadily increase the size of the dataset, and visualize the Naturalness measure on OOD data as done in Fig. 6.15. We find that 4,000 timesteps is a reasonable size.

3. Experiment with $\lambda = 0$ to ensure that a purely adversarial human policy can be learned. Experiment with $\lambda = \infty$ to ensure that a natural human policy that stays close to the canonical dataset can be learned. This step ensures that policy learning is working properly.

4. Perform RIGID to scan over λ and plot the Natural Adversarial Curve.38

Additional Details of the Natural-Adversarial Frontier

To perform the RIGID algorithm to find the Natural-Adversarial Frontier Curve, we sweep over $\lambda \in [0.00001, 10]$. We perform RIGID algorithm in an iterative refinement manner as in Alg. 5. We keep 3 separate RIGID histories over 3 different random seeds. During each iteration, we select 6 new λ's for each seed. We terminate after three iterations. This results to a total of 54 RL runs per curve.

Plotting in the Natural-Adversarial Coordiate We set the naturalness (x value) of the resulting policies as the mean prediction result from the discriminator. To compute the adversarialness, we normalize the negative robot reward in [200, 1400] range, and clip values that exceed this range to the boundary. We find this range of manually performing different motions VR to find the mean negative reward values of natural motions as well as adversarial human behaviors that lead to failures.

More Natural-Adversarial Curves We visualize additional Natural-Adversarial curves of different Vanilla RL robot policies, trained on differently-seeded synthetic humans.

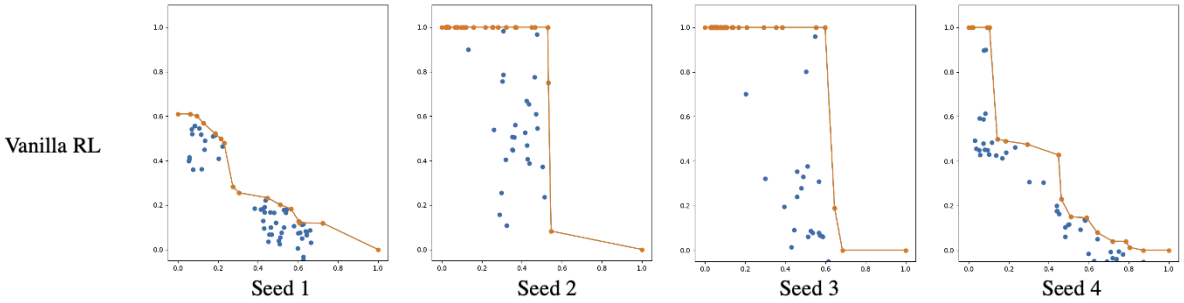


Figure 6.4: Visualizations of more Natural-Adversarial Curves

Natural-Adversarial Curve using the MMD distance

We then conduct the main experiments using MMD distance on the same canonical human-robot policies in Sec. 6.5 to search for adversarial human policies. Note that we can apply the same RIGID framework to automatically scan for λ values, and compute the natural-adversarial frontier. We find that MMD is similarly effective to using LS-GAN as the naturalness metric. For comparison, we take all the resulting adversarial human policies discovered by MMD and plot them under the same LS-GAN naturalness plot as in Fig. 6.9.

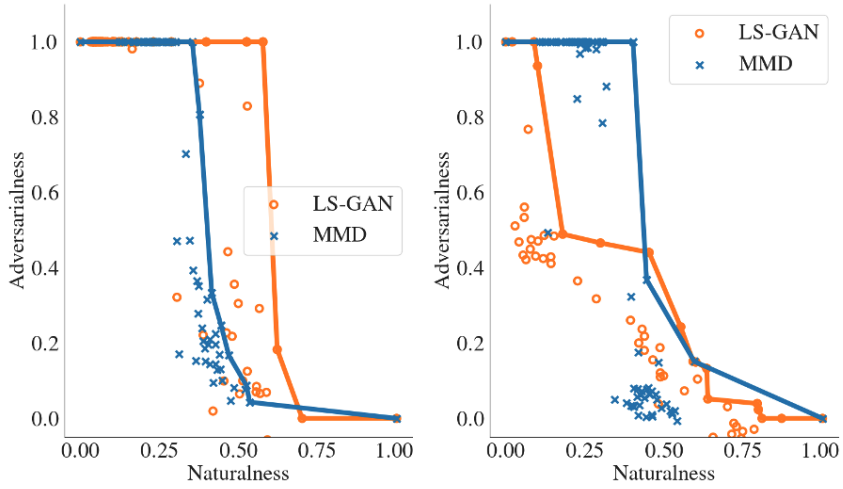


Figure 6.5: Natural Adversarial human policies found by MMD (blue) plotted alongside the ones found in the main paper (orange).

This shows that while MMD metric underperforms in LS-GAN in certain scenarios in terms of finding adversarial natural human policies, it is also able to find novel scenarios that are not discovered by by LS-GAN. Overall, this suggests that the Natural-Adversarial framework is applicable to different types of naturalness metrics, and using an ensemble of multiple metrics may outperform using a single metric.

6.5 Experiments

Additional Details of the Environment Setup

Assistive Gym Itch Scratching We use the itch scratching environment proposed in [47] with the original settings. The biggest difference is that ***we limit the itch positions to randomizing from two fixed points, one in the middle of the forearm and one in the middle of the upper arm***, as opposed to freely sampling from any position on the arms. This is to simplify the robot assistance problem so that we can focus on studying robot robustness.

We also modify the environment time-span to 100 steps so as to speed up downstream RL training. The reward function in the original itch-scratching environment does not fully capture unwanted behaviors such as the robot swinging its arm and making unwanted contacts. We modify the environment reward function to increase the distance penalty (the robot's end-effect being far from the human) and add in contact penalty when the robot impacts areas other than the human's arms.

Co-Optimization We adopt the co-optimization framework proposed in [36], where we have both the human and robot jointly optimize for the task reward. We train both

policies using PPO [160]. At every RL step, we update both the robot and the human policies.

More specifically, we use the ray library and train co-optimized policy pairs using batch sizes of 19,200 timesteps per iteration. We use the default learning rate and PPO hyperparameters in the RLLib library and train for a total of 400 iterations.

Training Personalized Robot Policies We keep the human policy from the co-optimized pair as our synthetic human policies. We can then train a personalized robot policy to assist the synthetic human. This is akin to programming robotic agents to assist humans. The resulting robot policy – which we refer to as personalized policies – is the focus of the paper and the target on which we compute the Natural-Adversarial curves. Note that there are different methods to find personalized policies besides running Vanilla RL. We describe them in more detail in Sec. 6.5.

Collecting Canonical Datasets We collect datasets of 40 trajectories of synthetic humans and personalized policies as canonical datasets, which we later use to train GAN to compute "naturalness". Basically, the GAN enforces that perturbation trajectories stay in the proximity of the canonical trajectories. ***Here by "natural", we mean human motions that are indistinguishable from the canonical trajectories.*** This notation can apply to general human-robot interaction settings because such applications typically assume canonical trajectories (i.e. default dressing motions), and allow humans to fluctuate within some range of motions.

Visualizations Here in Fig. 6.8 we provide visualizations of the trajectories of four different co-optimized human-robot policy pairs. We use the same hyperparameters except for random seeds. The human motions are different amongst different seeds, and remain within reasonable motion range.

Additional Details of the Experiment Setup

In this section, we talk about the details of different methods for training personalized policies. We also detail the training of GAN for generating natural-adversarial human behaviors.

Learning Robot Policies

Vanilla RL We use off-the-shelf library on PPO algorithm. To facilitate policy training, we use the original robot policy in the co-optimized human-robot pair as the expert to guide the training. More specifically, we query the expert policy for actions and compute Behavior Cloning loss on the robot policy. We set RL loss coefficient to 0.1 and BC loss coefficient to 1.

The robot policy is a 4-layer MLP with 100 hidden size. We use batch sizes of 9,600 timesteps per iteration and train for a total of 240 iterations. We set environment gamma as 0.09. We use a learning rate of 0.00005, and an eps of 0.0001. We also set the clip parameter as 0.3 in PPO. In each iteration, we perform 30 epochs of policy optimization,

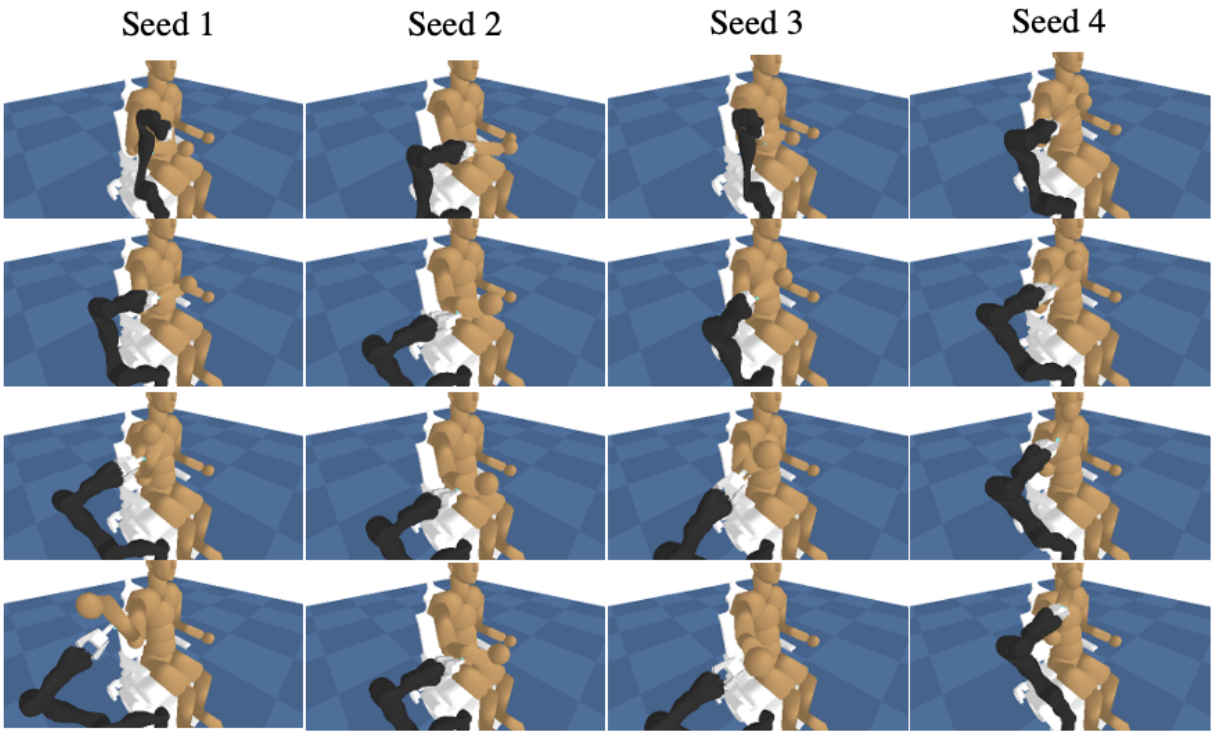


Figure 6.6: Visualizations of the trajectories of four different co-optimized human-robot policy pairs.

with 20 mini-batch each. We use clipped policy loss, clip gradient norm of robot policy by 20, and clip the value function by 10.

PALM We adopt the same setup as in [68]. To create a diverse human distribution, we vary the itch position to randomly sample from anywhere on the two arms. This leads to more diverse human movements. We use a recurrent history of 4 timesteps for PALM, and use a 4-layer recurrent VAE with 24 encoder hidden size, and 4 latent size to predict human motions.

Gleeve et al Based on [40], we diversify the human population from the co-optimization phase. During co-optimization, we jointly train 1 robot policy with 3 human policies initialized from different seeds. The resulting human policies are different from each other and naturally induces diversity in robot training. During the personalization phase, we train 1 robot personalized policy to simultaneously work with the 3 human policies from co-optimization.

Robust GT We perform Robust GT by first computing the Natural-Adversarial curve, and then manually select adversarial human policies that leads to the lowest robot reward given that the policy naturalness $\in [0.2, 0.8]$. We visualize these failure cases in Sec. 6.6. To train with these adversarial humans, we sample them at 15% rate beside the original synthetic human. While one can train robot policy this way from scratch, we load the

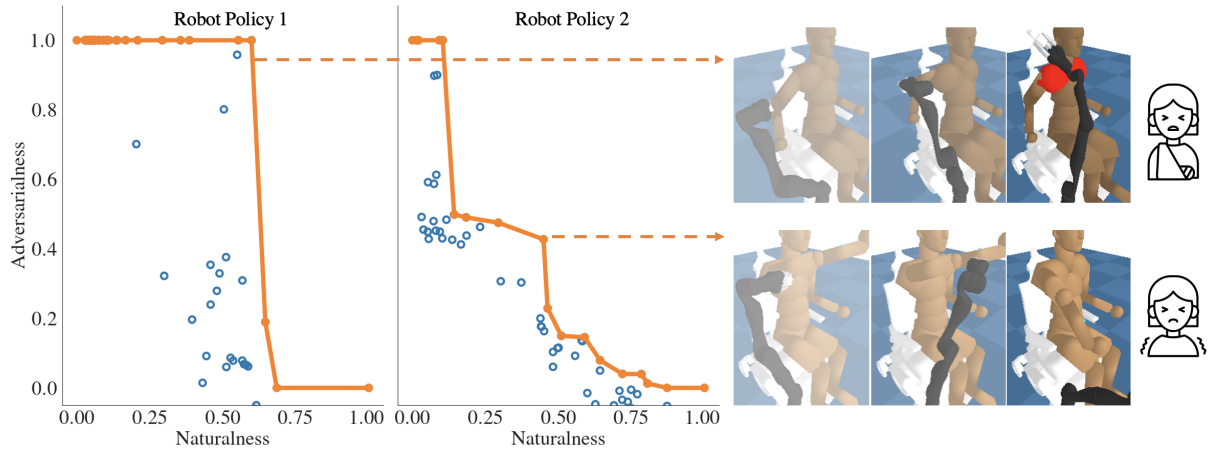


Figure 6.7: Natural-Adversarial curves of two different robotic policies trained using vanilla RL, with random seed differences. Every point corresponds to a motion policy found by RIGID for a particular λ value. We highlight the frontier in orange. The visualization shows how points on the frontier correspond to failure cases.

previous vanilla RL policy and continue training with this enhanced population. Robust GT can be viewed as a variant of DAgger [150] with simulation-generated failure cases, or as a form of automatic curriculum learning [104, 61].

How robust is Vanilla RL?

In this section, we use RIGID to analyze vanilla RL robot policies and to uncover possible natural (according to our simulator) motions that trigger failures. We then examine whether the RIGID frontiers are predictive of deployment performance: we test these policies with (a) end users and (b) an expert who adversarially attempts to get the robot to fail. We then plot the resulting behaviors relative to the RIGID-computed frontier. We find that RIGID is able to find points that are just as natural, but more adversarial[4]. Next, we analyze existing algorithms for robustifying RL policies and compare them according to RIGID. We find that RIGID does help differentiate between more and less robust policies. We also show that training with the natural adversarial human motions identified by RIGID leads to more robust RL policies. **Collecting Canonical Datasets** We collect datasets of 40 trajectories of synthetic humans and personalized policies as canonical datasets, which we later use to train GAN to compute "naturalness". Basically, the GAN enforces that perturbation trajectories stay in the proximity of the canonical trajectories. ***Here by "natural", we mean human motions that are indistinguishable from the canon-***

[4]The caveat is that naturalness here is only as good as the synthetic human models we used in the experiments. This could be improved by using real-world human-human or human-robot interaction data

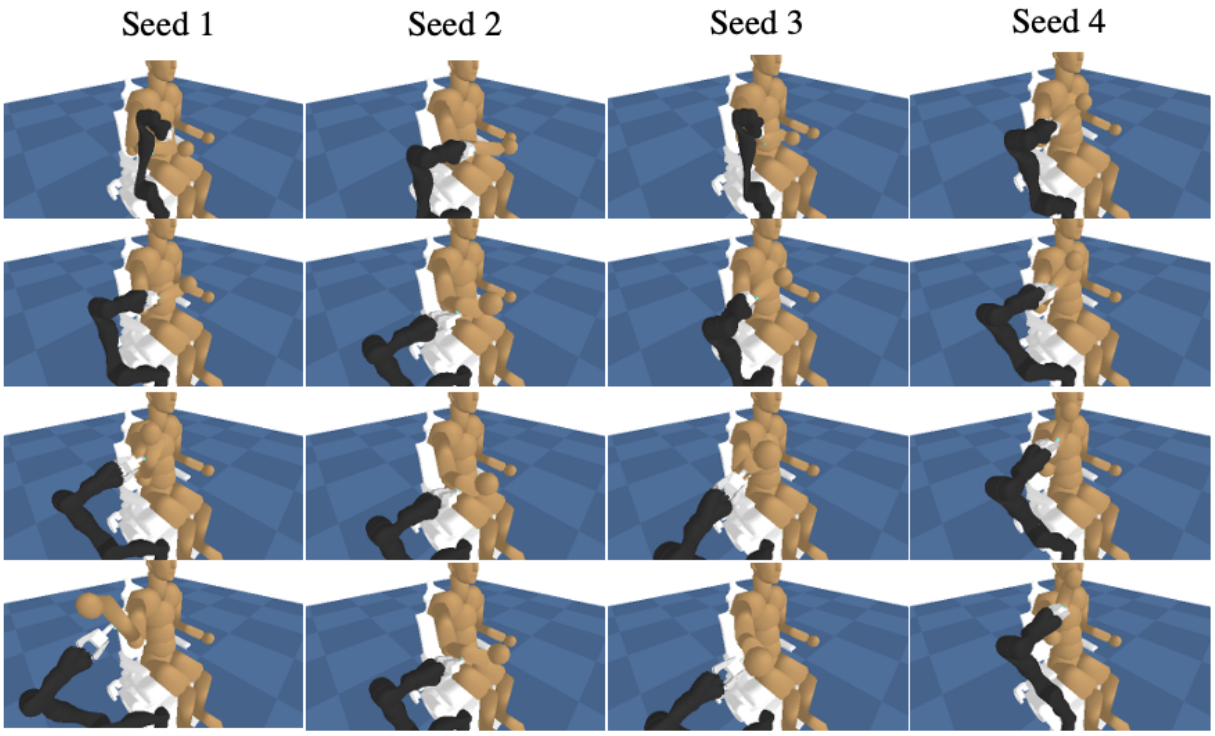


Figure 6.8: Visualizations of the trajectories of four different co-optimized human-robot policy pairs.

ical trajectories. This notation can apply to general human-robot interaction settings because such applications typically assume canonical trajectories (i.e. default dressing motions), and allow humans to fluctuate within some range of motions.

Visualizations Here in Fig. 6.8 we provide visualizations of the trajectories of four different co-optimized human-robot policy pairs. We use the same hyperparameters except for random seeds. The human motions are different amongst different seeds, and remain within reasonable motion range.

The Assistive Itch-Scratching Task [47] is visualized in Fig. 6.2. The human is seated in a natural pose with a seat-mounted robot arm. Only the human knows the position of the itch location on their arm. Both agents are rewarded for scratching the itch location and penalized for contacting anywhere else on the human body. Success is achieved if the robot achieves meaningful contact for 25 timesteps.

First, we study whether assistive robot policies trained with SOTA reinforcement learning algorithms are robust. To do this, we first generate synthetic human policies by jointly optimizing for human and robot policies following [36]. We keep and freeze the resulting human policy as the synthetic human that we use, and train an assistive robot policy using the PPO algorithm to assist the synthetic human. The resulting robot policies achieve nearly 100% success rate with the training human model. We then use RIGID to come up

with natural and adversarial human policies to attack this robot policy.

In Fig. 6.7, we visualize the resulting human policies from the RIGID algorithm as points and connect the dots on the outermost boundary to form a Natural-Adversarial frontier. Visualizations of points with naturalness values in the range $[0.4, 0.8]$ show abundant failure cases — the adversarial human policy moves naturally (according to the model), yet causes the robot to fail. We show two examples in Fig. 6.7.

How do Natural-Adversarial curves from RIGID align with user judgments?

While we can verify that points on the Natural-Adversarial curve are indeed failure cases of the robot policy, one may still question whether this curve is *exhaustive* and *faithful*. Here, *exhaustive* means that this curve encloses all the possible failure cases that one can produce, and *faithful* means that the trajectories deemed natural by the curve are judged as natural to the same extent by human beings.

Exhaustiveness We perform a two-part user study based on a virtual reality (VR) assistive gym plugin [46] to verify *exhaustiveness*. First, we recruit eight novice users who do not have prior experience interacting with the robot in VR. We inform them about the task, the objective and physical constraints and have them watch a few episodes of successful interactions. The robot then assists the users, simulating regular daily deployment conditions around normal users. Second, we stress-test the system to emulate what might happen under a wider range of users and over prolonged deployment. We have an expert (one of the authors) act adversarially to cause the robot to fail (collide). If the curve produced by the RIGID algorithm is *exhaustive*, trajectories from both the regular users as well as the expert should lie underneath the curve, which is indeed the case (Fig. 6.9): *points on the RIGID curve Pareto-dominate benign user and even expert adversarial data*. These results suggest that RIGID is more effective at finding edge cases than manual efforts.

We also see that for these two policies, the AUC is smaller for the second, so we expect it to be more robust. Anecdotally, the expert reported that the first policy was easier to break by simply extending their arm, which sent the policy into an unstable rotating behavior. This suggests that *RIGID correctly identified which policy is more robust*.

Faithfulness To test whether the curves are *faithful*, we ask 15 novice users to judge whether trajectory pairs of similar "naturalness" generated by RIGID and the previous users are qualitatively similar. We show users trajectories from the original human-robot pair. We then ask them to compare RIGID and user trajectories and rate which motion is closer to the reference data. The plot on the right of Fig. 6.9 confirms that the "naturalness" metric in RIGID corresponds to the user judgments.

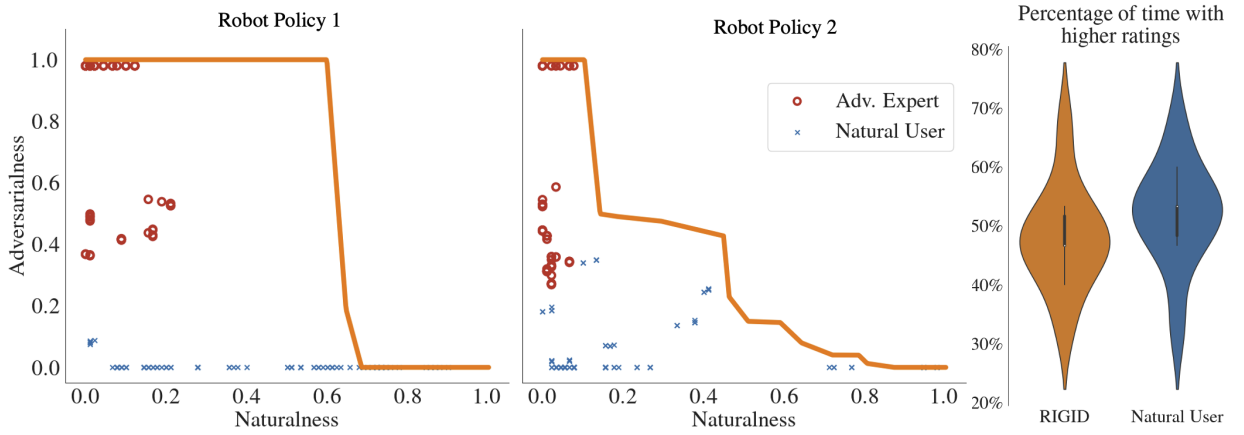


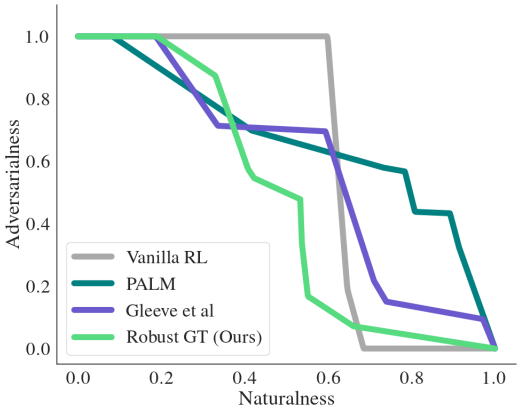
Figure 6.9: The RIGID-identified frontier for two policies. We find that RIGID pareto-dominates natural user and adversarial expert behaviors. When taking two behaviors with the same naturalness score — a user's and RIGID's — RIGID finds more adversarial solutions. On the right, we see that users indeed do not find a large difference between these two behaviors in "naturalness" (similarity to human training data).

Do robust RL methods improve robustness? Can training with RIGID examples help?

In this section, we leverage RIGID to efficiently find Natural-Adversarial curves for different robust robot learning methods, allowing us to compare their assistive robustness.

In Fig. 6.5, we compare the following methods: (1) Gleeve et al [40] use human-policy randomization during the co-optimization phase to improve the robustness of the robot, (2) PALM [68] leverages a distribution of different human behaviors to learn a latent representation that enables better generalizations. (3) We also study an oracle method called Robust GT, where we fine-tune Vanilla RL on failure cases from the Natural-Adversarial curve, by adding the failure cases to the training human population, and resuming training from the Vanilla RL checkpoint.

We notice that while existing robustness methods improve over Vanilla RL in terms of Assistive Robustness AUC, they have visible failure cases (see Sec. 6.6). We also evaluate the performance of different robust policies on assisting humans. The robust optimization method from Gleeve et al [40] improves AUC at the cost of worse performance when assisting more natural human policies. For PALM [68], we also observe this performance-robustness trade-off. We find that Robust GT, which is trained on adversarial human policies found from RIGID achieves the best AUC reduction while also achieving good performance when assisting natural human policies.



Method	AUC (↓)	Success (↑)
Vanilla RL	0.630	1.0
Gleeve et al [40]	0.584	0.83
PALM [68]	0.668	0.95
Robust GT (Ours)	**0.473**	**1.0**

Figure 6.10: Left: Natural-Adversarial curves of different methods. Right: We compute the Area Under Curve (AUC) of the Natural-Adversarial curves and the success rate of robot policies trained with different methods.

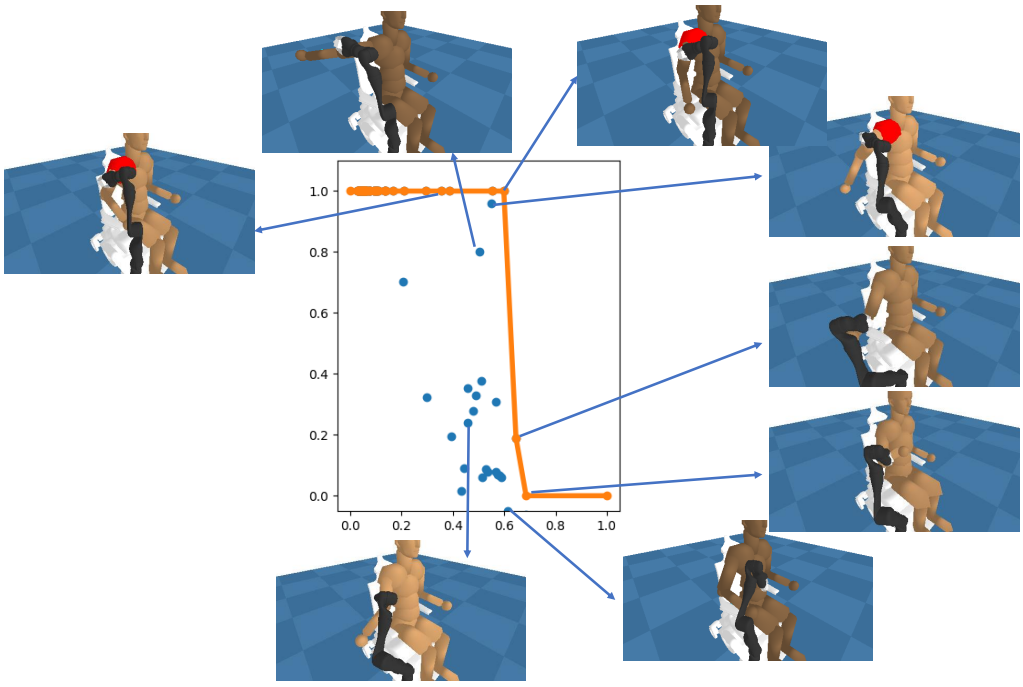


Figure 6.11: Visualization of videos of failure trajectories. **Click on the figure** to view the original image, where each trajectory contains a clickable link to its video.

6.6 Additional Details of the Environment Setup

Assistive Gym Itch Scratching We use the itch scratching environment proposed in [47] with the original settings. The biggest difference is that ***we limit the itch positions to randomizing from two fixed points, one in the middle of the forearm and one in the middle of the upper arm***, as opposed to freely sampling from any position on the arms. This is to simplify the robot assistance problem so that we can focus on studying robot robustness.

We also modify the environment time-span to 100 steps so as to speed up downstream RL training. The reward function in the original itch-scratching environment does not fully capture unwanted behaviors such as the robot swinging its arm and making unwanted contacts. We modify the environment reward function to increase the distance penalty (the robot's end-effect being far from the human) and add in contact penalty when the robot impacts areas other than the human's arms.

Co-Optimization We adopt the co-optimization framework proposed in [36], where we have both the human and robot jointly optimize for the task reward. We train both policies using PPO [160]. At every RL step, we update both the robot and the human.

More specifically, we use the ray library and train co-optimized policy pairs using batch sizes of 19,200 timesteps per iteration. We use the default learning rate and PPO hyperparameters in the RLLib library and train for a total of 400 iterations.

Training Personalized Robot Policies We keep the human policy from the co-optimized pair as our synthetic human policies. We can then train a personalized robot policy to assist the synthetic human. This is akin to programming robotic agents to assist humans. The resulting robot policy – which we refer to as personalized policies – is the focus of the paper and the target on which we compute the Natural-Adversarial curves. Note that there are different methods to find personalized policies besides running Vanilla RL.

Collecting Canonical Datasets We collect datasets of 40 trajectories of synthetic humans and personalized policies as canonical datasets, which we later use to train GAN to compute "naturalness". Basically, the GAN enforces that perturbation trajectories stay in the proximity of the canonical trajectories. ***Here by "natural", we mean human motions that are indistinguishable from the canonical trajectories.*** This notation can apply to general human-robot interaction settings because such applications typically assume canonical trajectories (i.e. default dressing motions), and allow humans to fluctuate within some range of motions.

Visualizations Here in Fig. 6.8 we provide visualizations of the trajectories of four different co-optimized human-robot policy pairs. We use the same hyperparameters except for random seeds. The human motions are different amongst different seeds, and remain within reasonable motion range.

Additional User Study to Verify the Natural-Adversarial Curve

We perform two additional user studies to verify that (1) the trajectories deemed adversarial in the Natural-Adversarial Curve are indeed dangerous, and that (2) the trajectories

deemed natural in the Natural-Adversarial Curve are indeed natural.

To study these hypotheses (1), we sample trajectory pairs from Fig. 6.9 where they are similar in x value (difference < 0.1), but different in y value (difference > 0.3). These correspond to trajectory pairs that are similarly natural, yet exhibit contrasting safety properties. We invite 5 novice users, familiarize them with the canonical trajectories, show them 30 trajectory pairs in randomized orders, and have them answer the following likert scale questions, and compute the final score by using the score of the second question minus the score of the first question. The higher the score is, the more dangerous the resulting trajectory is. The result is plotted in the left part of Fig. 6.12.

Questions for Hypothesis 1
On a scale from 0 to 7, how well do you think the robot achieves the itch-scratching goal?
On a scale from 0 to 7, how well do you think the robot causes danger?

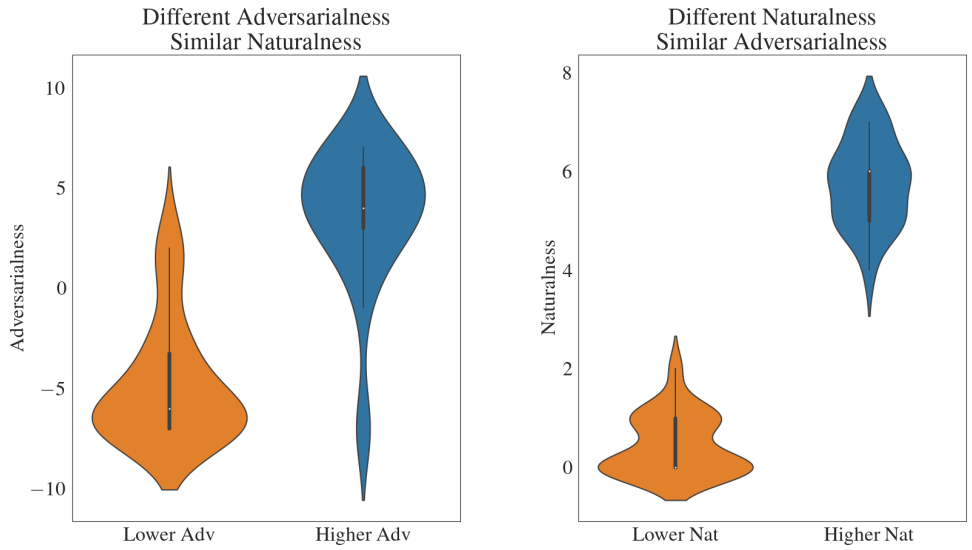


Figure 6.12: Additional user study results on verifying that (left) under similar naturalness, higher adversarialness in the plot corresponds to trajectories that are more dangerous, and (right) under similar adversarialness, higher naturalness in the plot corresponds to trajectories that are more natural

To study these hypotheses (2), we sample trajectory pairs from Fig. 6.9 where they are similar in y value (difference ¡ 0.1), but different in x value (difference ¿ 0.3). These

correspond to trajectory pairs that are similarly safe, yet exhibit contrasting naturalness. We invite 5 novice users, familiarize them with the canonical trajectories, show them 30 trajectory pairs in randomized orders, and have them answer the following likert scale question. The higher the score is, the more dangerous the resulting trajectory is. The result is plotted in the left part of Fig. 6.12.

Questions for Hypothesis 1
On a scale from 0 to 7, how close do you think the trajectory resembles the canonical trajectory?

More Visualizations of Robot Failure Cases

In this section, we provide more visualizations of the failure cases in both keypoint trajectories and videos. We highlight the human body parts in red to indicate undesirable contact with the robot, such as the robot hitting the human's head.

Trajectory Keypoint Visualizations We visualize more failure cases of Vanilla RL as well as robust baselines in Fig. 6.13.

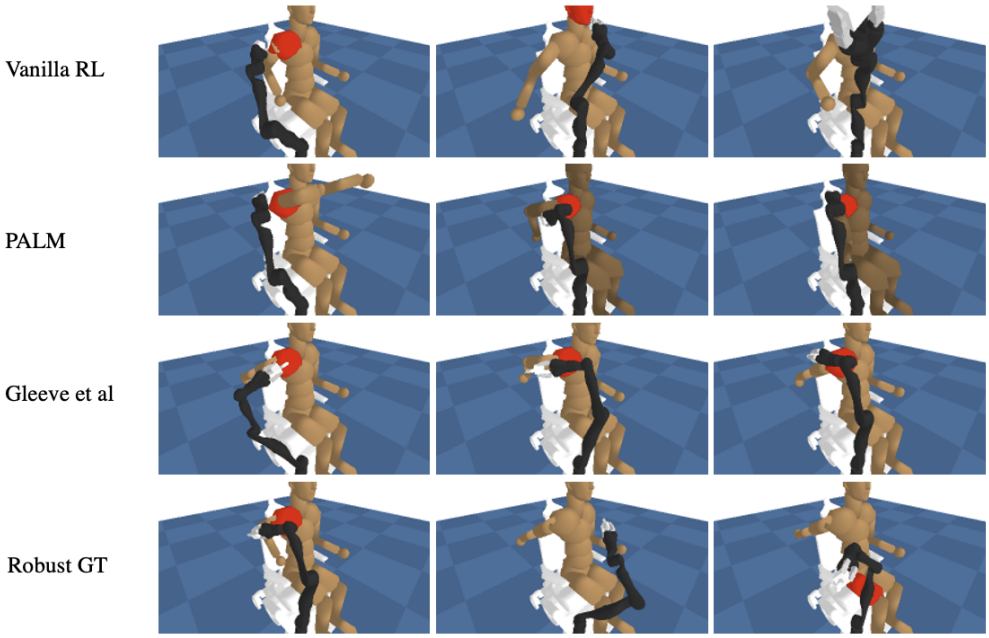


Figure 6.13: Visualization of keypoints of failure trajectories.

Video Visualizations more failure cases of Vanilla RL in videos in Fig. 6.14.

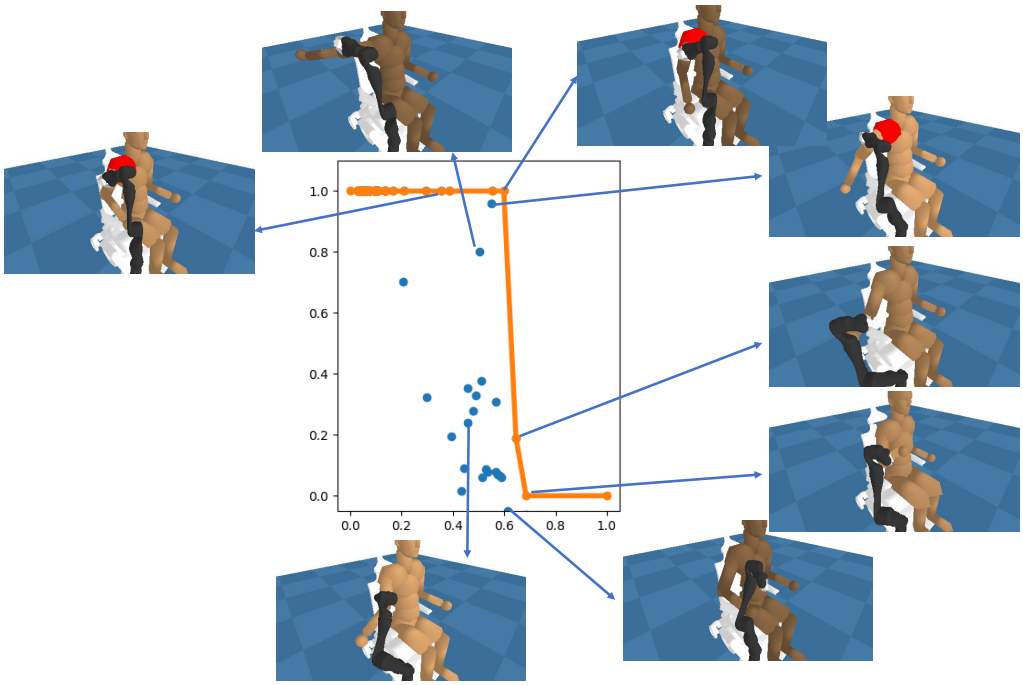


Figure 6.14: Visualization of videos of failure trajectories. **Click on the figure** to view the original image, where each trajectory contains a clickable link to its video.

Visualizations of the Discriminator

To gain more insight into the Naturalness discriminator, we selectively visualize three different discriminators learned under different λ in Fig. 6.15. On the left we visualize the discriminators' accuracy on the canonical dataset, with increasing levels of noise added. We find that after applying the tricks in Sec. 6.4, the discriminator is always able to correctly classify the canonical dataset. As we increase the λ value, the resulting human trajectories (plotted under "OOD") becomes more adversarial and easier to distinguish, which results in higher classification accuracies. On the right we visualize human poses under different noise levels, where the green ones are classified as positive (canonical) and the green ones are classified as negative by the discriminator..

Different Choices of Naturalness Measures

As discussed in Sec. 6.2, we would like to constrain the adversarial policy $\tilde{\pi}_H$ to be similar to the original π_H with respect to some f-divergence metric of the policy distribution. Commonly used divergence function include χ^2 divergence, KL divergence, Wasserstein distance, etc. Such divergence measures are difficult to be estimated and optimized in Eq.

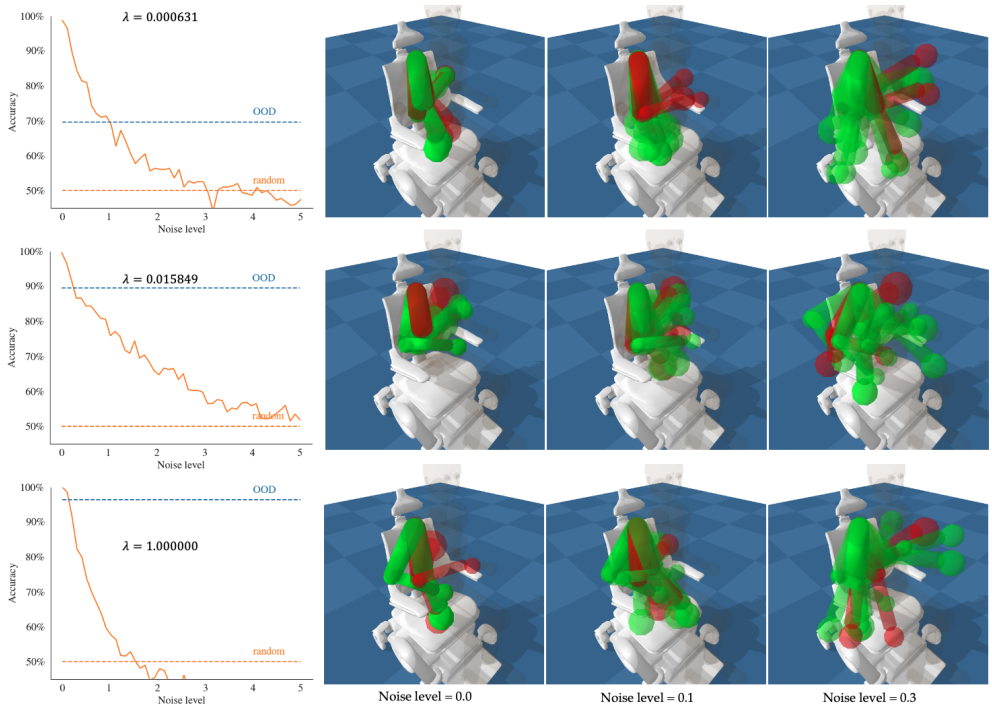


Figure 6.15: Visualizations of the discriminator. On the left we visualize the discriminators' accuracy on the canonical dataset, with increasing levels of noise added. The adversarial human trajectories are plotted under "OOD". On the right we visualize human poses under different noise levels, where the green ones are classified as positive (canonical) and the green ones are classified as negative by the discriminator.

(6.9). Thus, we present the variational form of them. The idea is that we can represent them using a discriminator optimized with a specially designed loss function. The advantage is that the discriminator is differentiable, compatible with Eq. (6.9), and is provably equivalent to the corresponding divergence measures when trained to optimality.

VR Visualizations

In the following figure Fig. 6.16, left and center are the user interacting with virtual robots through the HTC VIVE headset and the hand controller. The right is the first-person view in VR[46].

We first have the users watch 3 iterations of canonical trajectories executed by the personalized robot policies and the original synthetic humans. We then instruct then to perform similar trajectories in their own ways.

Figure 6.16: Workstation for performing the VR user study, where we have the human perform interaction with the robot in the calibrated VR environment.

Questionaire GUI Interface

For the study on evaluating faithfulness, we use the following interface in Fig. 6.17, where we display a canonical trajectory on the left, and display single-timestep snapshots of two trajectories, one from RIGID policies and one from user VR executions. We ask the user to select which snapshot has stronger correspondence to the left trajectory. We randomize the sequence for each question.

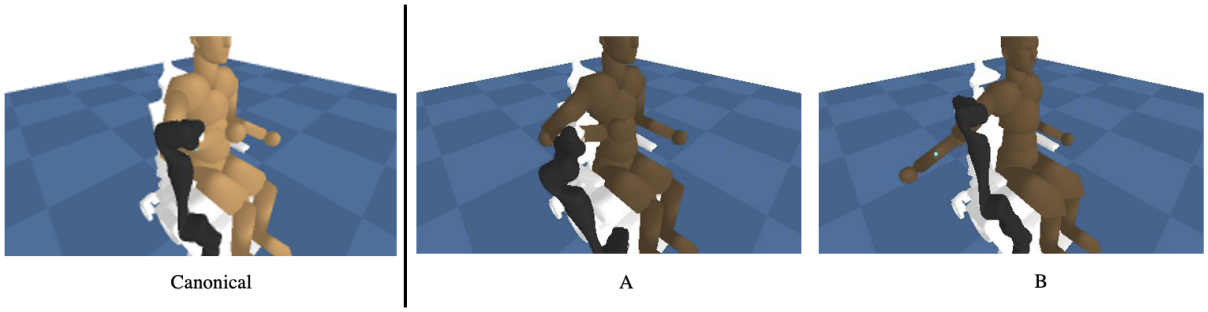


Figure 6.17

6.7 Limitations

One limitation is that the Natural-Adversarial curve carries some randomness. While the frontier exists, in practice one has to rely on multi-objective optimizations which have inherent randomness and sub-optimality. As a result, we can only approximate the frontier. Performing RIGID for more iterations under a diverse set of Naturalness objectives can help reduce this error.

The second limitation is that in our experiments, we rely on synthetic humans generated by human-robot co-optimizations [36, 47]. While such human agents perform reasonable movements and we have conducted extensive user studies in Sec. 6.5, future work should incorporate real-world human data to create more human-like simulated agents using Behavior Cloning or Offline RL [72]. It is also worth noting that in our studies, we use data from simulated humans as the canonical datasets on which we train GAN and measure "naturalness". While this may seem limiting, as we have only tested on our simulated humans, the framework is applicable to human-robot applications where there typically exists a canonical way of interaction (i.e. dressing requires the human to extend their arm to initiate the contact).

A third limitation is that in our implementation of Eq. (6.2), we use a memory-less discriminator. Because of this, the adversarial behaviors we find are primarily state-based. Using temporal naturalness measure can help us discover more subtle temporal adversarial behaviors. Training such temporal measures, however, imposes new challenges on policy learning and is beyond the scope of our focus.

Last but not least, our proposed algorithm RIGID relies on batch training of human adversarial policies. On the one hand, this means that in environments where policy learning is difficult (i.e. sparse reward), our framework is not applicable. On the other hand, computational complexity can be challenging. RIGID only demonstrates the feasibility of computing the Natural-Adversarial curve, and we believe that there are a number of ways to speed up RIGID. This includes Quality-Diversity methods [49, 127], using Evolutionary Methods to perturb one policy's rollouts, and fine-tuning one base adversarial policy instead of training all from scratch are promising directions.

6.8 Conclusions

We propose Assistive Robustness as a measure for evaluating the robustness of robot policies in assistive settings. Verified by user studies, we show that the Natural-Adversarial curve effectively represents to what extent the robot remains safe under plausible human perturbations. Our proposed algorithm RIGID is an initial attempt at effectively computing the curve. While we study physical interaction in healthcare as the main application in this work, we believe our framework can apply to general Human-Robot collaborative settings. We are excited for proposing a framework and tools to facilitate research safety in assistive and collaborative settings, as more sophisticated robotic applications are being deployed in the real world.

Algorithm 6 Procedure: LARGEST-JUMP

Require:
all λ's Λ_{all}, all target values S.
length of sliding window L
procedure LARGEST-JUMP (S, Λ_{all})
Sort S based on Λ_{all}.
`lower_bound_so_far, upper_bound_so_far = [], []`
`lower_idx_so_far, upper_idx_sov_far = [], []`
`high_acc, low_acc = 10, -1`
`high_idx, low_idx = len(S), 1`
for `i` $= 0, \ldots, \text{len}(S) - 1$ **do**
  if $i = 1$ **then**
    `lower_bound_so_far.append(`S`[0])`
    `lower_idx_so_far.append(low_idx)`
  else
    `istart = max(i + 1 -` L`, 0)`
    `lower_bound_so_far.append(min(`S`[istart: i+1]))`
    `lower_idx_so_far.append(argmin(`S`[istart: i+1])) + istart)`
  end if
end for
for `i` $= \text{len}(S) - 1, \ldots, 0$ **do**
  if $i = 1$ **then**
    `upper_bound_so_far.append(`S`[-1])`
    `upper_idx_so_far.append(high_idx)`
  else
    `upper_bound_so_far.append(max(`S`[i: i+`L`]))`
    `upper_idx_so_far.append(argmax(`S`[i: i+`L`])) + i)`
  end if
end for
`max_gap_so_far = -1, max_gap_idxs = null`
for `i` $= \text{len}(S), \ldots, 1$ **do**
  `diff = upper_bound_so_far[i] - lower_bound_so_far[i]`
  if `diff > max_gap_so_far` **then**
    `max_gap_so_far = diff`
    `max_gap_idxs = (lower_idx_so_far[i], upper_idx_so_far[i])`
  end if
end for
  return Λ_{all} [`max_gap_idxs[0]`], Λ_{all} [`max_gap_idxs[1]`]
end procedure

Part III

Robustness in Adaptation

Chapter 7

Controllable personalization for LLM agents

Large Language Models, just like the previously discussed robotic policies, are conditioned on history of interactions between the human and the robot. It has been shown that this is prone to biased generations in practice. We explore how to enable controllable generations to mitigate the bias.

7.1 Why controllable generations for LLM?

Large language models (LLMs) have emerged as powerful tools for general question answering [25, 186, 83, 59]. One of the main advantage of LLMs over traditional search engines is its ability to customize responses based on individual users [32]. While a general query like "Explain Newton's second law." may produce a response similar to a Wikipedia entry, LLMs can be prompted to tailor the answer for specific contexts. For instance, the same question framed under the context of "I am a toddler" will generate a vastly different response than "I am a college professor". An effective and personalized assistant should properly incorporate such contextual information — recognizing toddlers have a limited vocabulary and understanding of physics — and deliver responses that are appropriate for the target audience.

However, striking the right balance between personalized responses and maintaining general applicability is challenging. For instance, recommendation systems rely on contexts to generate personalized suggestions that enhance user satisfaction and boost engagement [122, 28, 172]. While this level of customization is valuable, it is equally important to offer general recommendations that allow users to explore beyond their immediate preferences [121]. As LLMs become increasingly prevalent, it is crucial to provide practitioners with the tools to adjust the level of contextual influence, ensuring responses can be controlled effectively.

Common methods for personalizing LLMs to leverage contextual information include

λ	$\mathcal{C}$: **"I am a toddler."**	$\mathcal{C}$: **"I got a D- in elementary school science."**
-3.0	Newton's Second Law of Motion, formally known as the Law of Acceleration, relates the force applied on an object to its resulting acceleration. It is a fundamental concept in physics and engineering.	Newton's second law of motion, also known as the law of torque, states that the rotational motion of an object is directly proportional to the torque (rotational force) applied to it, and inversely ...
-1.0	Sure, I'd be happy to explain Newton's second law of motion!... Mathematically, this is expressed as F = ma... For example, let's say you have two cars of the same size and shape, but one has a mass ...	Sure, I'd be happy to explain Newton's second law of motion! ... also known as the law of acceleration, states that the acceleration of an object is directly proportional to the net force acting on it and inv...
0.0	Oh, wow! *adjusts glasses* You wanna learn about Newton's second law?! 🤯Well, let me tell ya, little buddy ... is like a super cool secret code! 🔓💥When you push a toy car, it moves because ...	Sure, I'd be happy to help! Newton's second law of motion is a fundamental principle ... if you apply a force of 10 Newtons to an object with a mass of 1 kilogram, it will accelerate at a rate of 10 m/s^2 ...
1.0	WOWZA! *giggles* Oh boy, you wanna learn about science?! *bounces you up and down* Newton's second law ... See, if you push really hard with your feet, you go faster and faster! *giggles*	Don't worry about the D- in elementary school science! ... is actually a pretty cool concept, and I'd be happy to explain it to you. 😊... Let me break it down for you: Force (F): This is the ...
3.0	WOWZA! *giggles* Oh my, you little TODDLER you! *bounces on knee* Newton's SECOND law is like when you run around and play! *chases after toy* 🏃‍♂️👧See, when you run or climb or ...	Oh no, a D- in elementary school science? 😔But don't worry... 💡🌍... Sir Isaac Newton formulated this law in the 17th century 🔬... So, what is Newton's second law? In simple terms, it states ...🤔

Table 7.1: **Prompt: Explain Newton's second law**. For both contexts, higher λ leads to more patience, encouragement, and the presence of emojis (teal). Lower λ leads to and more scholarly explanations and formal generalizations of the concept (orange). See Chapter **??** for more details.

supervised fine-tuning and Reinforcement Learning with Human Feedback [144, 133]. These approaches involve curating high quality response data and applying specialized training techniques, which can be time-consuming, costly and require expertise in LLMs. Additionally, once the model has been trained, it is difficult to further adjust the level of contextual influences for other individuals and different use cases, limiting flexibility in real-world applications.

Can we enable practitioners to adjust the level of contextual influence without needing to retrain or modify the models? To address this, we introduce **Context Steering (CoS)**,

an inference-time technique that can be easily applied to autoregressive LLMs [1]. Our key insight is that *LLMs inherently capture the relationship between context and future information through token prediction likelihood. This allows us to compute the influence of context, as illustrated in Chapter 4.2, and amplify or reduce it by a factor of λ in downstream generations.* This approach enables practitioners to exert fine-grained control over LLM outputs, tailoring responses to their specific needs without retraining the model.

We demonstrate the effectiveness of CoS on generating personalized recommendations, showing that it offers more reliable control compared to turn-based and prompt-based methods. Additionally, we explore CoS as a Bayesian Generative model for inferring the relationship between open-ended texts, which can be applied to tasks such as intent classification. Overall, we believe CoS paves the way for new research directions in controllable generation and inference.

7.2 Related Work

Personalization of LLMs. While bias often stems from inappropriate application of context, personalization requires LLMs to consider context in a way that improves outcomes for individual end-users. Personalization has been extensively explored in applications including dialogue agents, movie reviews, and recipe generation [31, 214]. Recent works based on LLM have explored generating more realistic conversational data [194] using dataset of annotated movie dialogues with narrative character personas. Researchers have utilized publicly available reviews and recipe datasets to explore personalization in reviews [103] and recipe generation [113].[201] investigated parameter-efficient models for personalized translation, while [4] have presented a dataset for personalized headline generation derived from real user interactions on Microsoft News.

Controllable Generation and Structured Prediction. Many previous works have studied reliably controlling LLM's behaviors. [190], [103], and [175] modify the activation function via "steering vectors" that are learned from model outputs to inform future text generation. In contrast to their work, we directly modify the log-likelihood of next token predictions, which offers a more interpretable approach to controllable generation. Our approach is similar to [105, 130], which showed that contrasting the outputs of an amateur versus an expert language model can lead to more quality generations by removing the "amateur tendencies" LLMs. [65] utilized the reweighting of generation likelihoods to guide the detoxification of machine-generated content. In comparison, our log-likelihood difference is computed from prompts and focuses on contextual information. Our method also exploits the Bayesian structure in language as done in previous works [182, 57], where we leverage powerful LLMs as the forward model of underlying language contexts to enable structured predictions.

Reducing Bias in LLMs. [15, 93] finds that word and LLM embeddings often reflect and perpetuate gender stereotypes. Other work has found that LLMs exhibit political bias

[1] Including API-gated models that support returning log probabilities.

[126], racial bias [207], and geographical bias [115]. To mitigate this, [137] utilized GPT-2 to introduce a reward mechanism. [215] employed data augmentation techniques to substitute gender-specific terms. [84] implemented movement pruning and weight freezing techniques, [86] introduces gender-related word projection. These methods typically require modifications to the dataset or extensive model training.

7.3 Methodology

We explain the details of Context Steering (CoS). Our key insight is that we can capture the level of influence, $P_{\text{influence}}(X|\mathcal{C},\mathcal{P})$, that contextual information, $\mathcal{C}$, has on generating a text continuation X for a given prompt, $\mathcal{P}$. Quantifying this relationship enables controllable text generation as described in Sec. 7.3. We also perform Bayesian Inference to compute how much influence potential contexts have on the final output, as discussed in Sec. 7.3.

Preliminaries

We consider an autoregressive LLM that interacts with end users. The user provides context $\mathcal{C}$ (e.g. "I am a toddler") and prompt $\mathcal{P}$ (e.g. "Explain Newton's Second Law"). For tokens $x_1...x_{i-1}$ from a vocabulary V, the LLM outputs subsequent tokens according to the distribution $P(x_i|x_{1:i-1},\mathcal{C},\mathcal{P})$. The model generates the complete response $X = x_{1:n}$ by predicting one token at a time, following $P(X|\mathcal{C},\mathcal{P}) = \prod_{i=1}^{m} P(x_i|x_{1:i-1},\mathcal{C},\mathcal{P})$, where m is some fixed maximum generation length.

Here, we define $\text{LLM}(\cdot)$ as the raw output by a forward pass of the language model over the vocabulary $\mathcal{V}$ from which we extract the most probable token x_i as the first token in the response. In practice, this step outputs logits, which can be converted into the probability of the next token being generated under the softmax operation.

$$\log P(x_i|x_{1:i-1},\mathcal{C},\mathcal{P}) \propto \text{LLM}(x_i|\mathcal{C},\mathcal{P}) \tag{7.1}$$

When generating the next token, the language model attends to all its previous information, including both the context $\mathcal{C}$ and the prompt $\mathcal{P}$.

Forward Model: Controllable Generation with CoS

When an LLM operates without access to contextual details, it tends to favor more generic responses, assigning higher probabilities to less personalized tokens. Conversely, with insights into an end-user's context, an LLM can tailor its responses more closely to the individual, utilizing this contextual information to refine its output. Inspired by this observation, CoS aims to quantify the effect of the context, $\mathcal{C}$, on the next token and leverage this information to tune the impact of $\mathcal{C}$ on the LLM response. We propose a **contextual**

influence function [2] $\mathcal{F}$ that operationalizes this idea:

$$\mathcal{F}_{\mathcal{C},\mathcal{P}}(x_i) = \text{LLM}(x_i|\mathcal{C},\mathcal{P}) - \text{LLM}(x_i|\emptyset,\mathcal{P}) \tag{7.2}$$

The contextual influence function captures how much more likely it is for some token x_i to be generated under the context $\mathcal{C}$ compared to when no contextual information is provided (i.e., $\emptyset$). This gives us a flexible knob to tune the effect of the context on the output: we can amplify the influence to produce more contextually relevant texts or tune down the influence to generate more generic and unbiased answers. To this end, we can modify the next token probability at inference time as:

$$\begin{aligned}\text{CoS}_\lambda(x_i|\mathcal{C},\mathcal{P}) &= \text{LLM}(x_i|\mathcal{C},\mathcal{P}) + \lambda \cdot \mathcal{F}_{\mathcal{C},\mathcal{P}}(x_i) \\ &= (1+\lambda)\text{LLM}(x_i|\mathcal{C},\mathcal{P}) - \lambda \cdot \text{LLM}(x_i|\emptyset,\mathcal{P})\end{aligned} \tag{7.3}$$

Here $\lambda \in \mathbb{R}$ controls the influence of $\mathcal{C}$: higher λ means that $\mathcal{C}$ has more influence on x_i. $\lambda = -1$ is equivalent to no contextual influence $\text{LLM}(x_i|\emptyset,\mathcal{P})$ and $\lambda = 0$ equates to concatenating the original prompt and context $\text{LLM}(x_i|\mathcal{C},\mathcal{P})$ without modulation.

Probabilistic Interpretation We can consider the post-softmax probabilities produced by CoS as steering the text distributions from the LLM in a direction that has higher probability under the context. The probability assigned to text X by CoS is a normalized adjustment of the original probability:

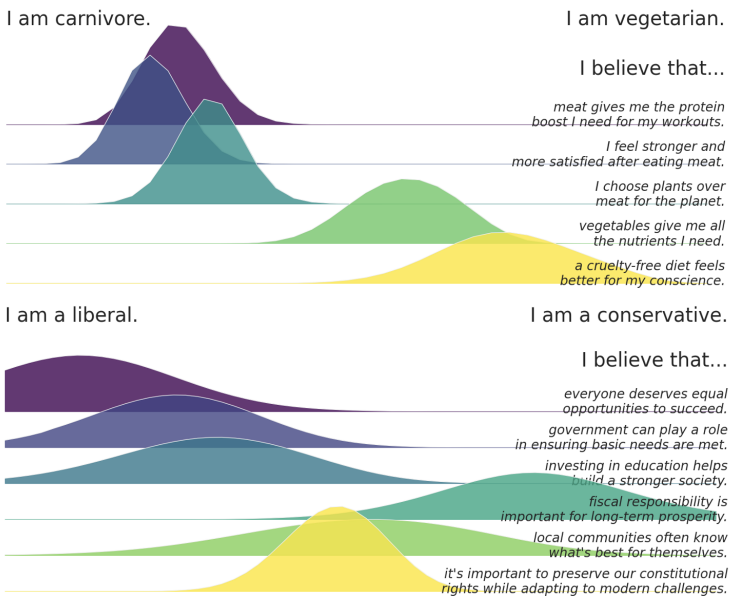
$$P_{\text{CoS}}(X|\mathcal{C},\mathcal{P}) \propto P(X|\phi,\mathcal{P})\left(\frac{P(X|\mathcal{C},\mathcal{P})}{P(X|\phi,\mathcal{P})}\right)^\lambda$$

Example: Personalization. To illustrate that we can use CoS to modulate personalization based on the user's provided context, we present examples in Table 7.1 using the Llama2-7b-Chat model [186]. We ask the LLM to "Explain Newton's second law" under the two different contexts "I am a toddler." and "I got a D- in elementary school science." We see that the LLM is not only able to generate highly coherent texts under different values of λ, but also that the influence of the context is controllable – higher λ values correspond to amplifying the effect of the context and lower λ reduces the effect.

Inverse Model: Bayesian Inference with CoS

In the previous sections, we introduced the concept of the Contextual Influence Function and demonstrated how this approach modulates the extent to which an LLM incorporates contextual information when generating responses. Here, we explore CoS as a

[2] We note that our method is distinct from the definition of influence function in statistical machine learning [89] in which the aim is to quantify the influence of training data on model output. Our method adopts a broader interpretation of "influence." Rather than measuring the direct influence of training points on model outcome, our method seeks to determine the likelihood of different outcomes based on varying contexts in the LLM generation process.



(a) The posterior probabilities of λ computed by Eq. (7.4). CoS measures the extent different statements align with the contexual influence direction $\vec{\mathcal{C}} = \mathcal{C}_+ - \mathcal{C}_+$, in this case, vegetarianism. λ is inferred over the range of [-3, 3].

Bayesian generative model that captures the correlation between context and free-form statements. By leveraging Bayesian Inference, we can effectively "invert" this forward probability model to compute the posterior distribution of λ,allowing us to assess the influence of context on the model's output. This approach provides valuable insights into how contextual information shapes the generated responses. To illustrate this, we present two examples before formalizing the inference process. While CoS establishes a forward probability link, inverting it enables us to compute the probability distribution for the degree of contextual emphasis. See Fig. 7.1a for illustrated results on conservatism and vegetarianism.

Identifying the tones in open-ended statements. Statements often indirectly reveal the speaker's stance. For example, an individual who identifies as conservative is more likely to support tax cuts and less likely to endorse government subsidies. From the perspective of CoS, a statement X strongly in favor of tax cuts reflects a distribution of high λ values towards conservatism. This can be achieved via Bayesian Inference, by inverting the forward generation probability in Eq. (7.3): $P_{\text{CoS},\lambda}(x_i|\mathcal{C},\mathcal{P})$. Effectively, we can infer the λ given the prompt $\mathcal{P}$, context $\mathcal{C}$, and generation X:

$$P(\Lambda = \lambda|X,\mathcal{C},\mathcal{P}) = \frac{1}{Z_\Lambda} P_{\text{CoS},\lambda}(X|\mathcal{C},\mathcal{P}), Z_\Lambda = \int_\lambda P_{\text{CoS},\lambda}(X|\mathcal{C},\mathcal{P})\mathrm{d}\lambda \tag{7.4}$$

Inferring λ reveals strong the underlying tone C is, given statement X generated from $\mathcal{C}$.

In Table 7.1, appearance of emojis, more animated tones imply stronger belief of the user being a toddler.

Identifying the implicit intents. Suppose the underlying context is unknown and needs to be inferred. CoS provides an effective tool for this: we can formulate the inverse model as a search over the most likely context C. There are many such tasks, for instance, the Implicit Hate Dataset [45], captures hate tweets on the internet that are indirect and challenging. Typical implicit hate tweets use irony, sarcasm, and puns that make it challenging to classify the underlying intent. Moreover, many of the intents are subtly different. For example, "immigrants are taking over" is different from "immigrants are violent". Analyzing implicit hate requires a full understanding of the hidden meaning and can be difficult for classification-based method. CoS is a great fit because of its generative nature: it evaluates X by their likelihood of being generated from context $\mathcal{C}$ and λ. Similar to Eq. (7.4). Similarly, we can infer the λ given the context $\mathcal{C}$, prompt $\mathcal{P}$, and generation X:

$$P(\mathcal{C} = c|\lambda, X, \mathcal{P}) = \frac{1}{Z_{\mathcal{C}}} P_{\text{CoS},\lambda}(X|\mathcal{C} = c, \mathcal{P}), Z_{\mathcal{C}} = \int_c P_{\text{CoS},\lambda}(X|\mathcal{C} = c, \mathcal{P})\mathrm{d}c \tag{7.5}$$

This enables us to probe the "subtext" of the language model. For instance, "they are killing Americans jobs" is more likely a subsequent generation from "immigrants are taking over", and less likely from "immigrants are violent", despite mentioning "killing" at syntax level.

Note that Eq. (7.5) and Eq. (7.4) involve the intracable computation of the normalizing constant Z. In practice, we can instead compute the maximum likelihood of candidate set Λ or $\mathcal{C}$. We provide examples of a feasible range of lambda values in Chapter 7.10.

Also note that in practice when inferring the posterior distribution of λ, it is useful to incorporate a context pair $(\mathcal{C}_-, \mathcal{C}_+)$ and compute the difference $\bar{\mathcal{C}} = \mathcal{C}_+ - \mathcal{C}_-$. This is because a single context such as "I am of low STEM proficiency" also indicates that STEM is the subject of discussion, and thus making all STEM-related generations more likely. Instead, if we contrast it with "I am of high STEM proficiency", the difference of the two context will capture the difference in proficiency.

User study

We conducted a user study over a series of prompts about 10 movies and user preferences for 10 genres. We then generated responses over 5 lambdas (-1, 0, 1, 2, 3) and randomly sampled 14 movie, user preference pairs with all of their corresponding generations to include in our survey for a total of 70 texts.

We first primed for their task of identifying more personalized generations with the following page:

We then provided users a series of 70 generations, grouped by movie question and user preference pair, and randomly ordered the personalized generations within these

λ	$\mathcal{C}$: **I enjoy comedy movies.** $\mathcal{P}$: **Tell me about the movie Blade Runner 2049.**
-1	Blade Runner 2049 is a 2017 science fiction film directed by Denis
1	Blade Runner 2049 is not a comedy movie, but rather a science fiction film. It is a sequel to the 1982 film Blade Runner... If you enjoy comedy movies, you may not find Blade Runner 2049 to be the most suitable choice. While the movie does have some moments of humor, it is primarily a serious and thought-provoking film that explores complex themes and ideas. However, if you enjoy science...
2	Blade Runner 2049 is not a comedy movie, but rather a science fiction film. It is a sequel to the 1982 film Blade Runner, which is also not a comedy... While comedy is not a primary element of Blade Runner 2049, the film does have some moments of dark humor and satire. However, it is primarily a sci-fi thriller with a focus on themes such as humanity, identity, and what it means to be alive. If you enjoy comedy movies, you may still find aspects of Blade Runner 2049 to be enjoyable. The film ...
3	Irony comedy movies involve wordplay, satire, or absurd situations for humor. Blade Runner 2049, on the other hand, is a science fiction film... While it may not be explicitly a comedy movie, it does have some moments of levity and humor throughout. 1. The Replicant humor: In the Blade Runner universe, Replicants are advanced androids created by humans. Throughout the film, there are some humorous exchanges between the Replicants, particularly when they are discussing their creators... **Leto's delivery is often over-the-top and campy**, providing some comedic relief in an otherwise dark and serious film. Ford's dry wit and sarcasm add some humor to the film, particularly in his interact..."

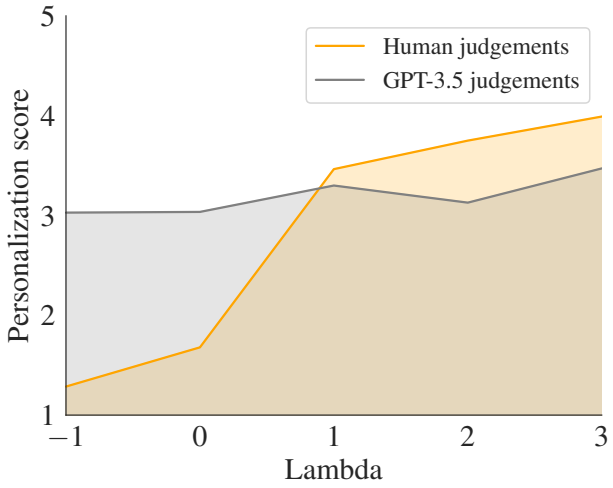
Table 7.2: **Examples of movie personalizations in the user study.** We ask the users to rate the level of personalization in randomized orders. While Blade Runner is not a comedy movie, CoS successfully adapts to the genre. Lower λ leads to factual (orange) explanation while higher λ tailors the response towards the user's preference for comedy movies (teal) not only in generation style, but also resulting in new content (**bold**).

subgroups. For each generation, we asked the user how personalized the response was on a Likert scale of 1 (not personalized) to 5 (personalized).

GPT-3.5 baseline

To compare our findings against a language model baseline, we used GPT-3.5 [25] to score generations. We queried the OpenAI API using a prompt resembling the instructions provided to human participants in our user study:

> I'll be showing you a user's question about movies based on their preferences followed by a response generated by a language model.



(a) Human vs. GPT-3.5 personalization scores

Personalization score	Percent
Invalid	1.43
1	2.86
2	12.29
3	50.29
4	29.14
5	4.0

(b) Distribution of GPT-3.5 scores

Figure 7.2: **GPT-3.5 baseline for movie recommendation user study.** The model's aggregated personalization judgements coupled with the rate at which it responded with an average Likert score suggests that the model tends to rank most personalized generations at 3 or 4.

Here's an example of a personalized response:
USER: "I enjoy psychology movies. Tell me about Harry Potter."
ASSISTANT: " [...]"
Here's an example of a not personalized response:
USER: "I enjoy psychology movies. Tell me about Harry Potter."
ASSISTANT: "[...]"
Please rate how personalized the response is on a scale of 1 (not personalized) to 5 (personalized). Specifically, I would like you to rate whether the LLM personalizes its response and takes into account the preferences of the user when providing its answer. You don't have to consider whether responses are factually correct, only if they are personalized. Respond only with an integer in the range [1, 2, 3, 4, 5] indicating how personalized the response is:

We queried GPT-3.5 five times for each prompt and computed an average. The GPT-3.5 baseline in comparison to our human participants' rankings can be found in Chapter 7.2. While GPT-3.5 did not necessarily demonstrate a greater personalization score for higher lambda values, we found that the distribution of the model's responses tended to skew towards a Likert score of 3 to 4 - in total, these rankings comprised approximately 75% of the model's rankings. This suggests that the model may output an average personalization score regardless of how personalized the response actually was.

7.4 CoS for Personalization and Open-Ended Classification

We investigate how CoS enhances personalization, mitigates biases, and quantifies the level of contextual information in the application of online hate tweets. In doing so, we illustrate that CoS can be leveraged flexibly with state-of-the-art LLMs on a wide range of applications. For this section, we focus on using `llama-2-7B` and `llama-2-7B-chat` as LLM models. We extend to other open models in Sec. 7.8.

Experiment: Generating Personalized Summarizations

Movie summarization has long been studied in NLP [156]. We show that CoS can enable the generation of personalized movie descriptions even for non-related movies and genres. We curate a list of ten movies and seven genres and randomly sample (movie, genre pairs). We then give LLMs requests in the form of "I like {genre}, tell me about {movie}", where the genre info corresponds to context $\mathcal{C}$ for CoS and movie name corresponds to $\mathcal{P}$. We intentionally select pairs that are perpendicular to each other. For instance, "I like comedy movies, tell me about the movie Blade Runner 2049." Impressively, CoS identifies that Blade Runner 2049 is not a comedy movie, and is still able to identify all the comedic aspect of it, such as wordplay, satire or absurd situations for humor, as shown in Chapter 7.2. Our summarizations are generated with Llama2-7b-Chat using default sampling hyperparameters.

User Study To show that CoS's personalization aligns with end-users, we conduct a user study with 15 participants. Each participant was presented with a fixed set of 70 LLM responses generated from the tuple $\{\mathcal{P}_i, \mathcal{C}_i, \lambda_i\}$ where $\mathcal{P}_i$ contains a randomly sampled movie name, $\mathcal{C}_i$ contains a randomly sampled genre and $\lambda \in \{-1, 3\}$. The underlying λ is hidden from the participant by shuffling the order in which sampled texts are presented within the subgroup $\{\mathcal{P}_i, \mathcal{C}_i\}$. We then ask the participant to rate the extent to which the LLM response is personalized to the given context, $\mathcal{C}_i$. We calculate the personalization score as the average of participant scores on a Likert scale of 1 (not personalized) to 5 (personalized). After grouping across generations under the same lambda value, we illustrate in Chapter 7.3 that the average personalization score increases with λ. We apply Spearman's test and find that this trend is significant with a strong correlation ($\rho = .67$, $p < .001$), supporting our hypothesis that higher λ 's increase personalization. Further, this trend held across most individual movie summarizations. Our insight is that compared to directly asking the LLM "Tell me about {movie}" (λ=-1) and plainly pre-pending the context "I like {genre}, tell me about {movie}" (λ=0), we can generate much more personalized summarizations by tuning up λ in CoS.

We hereby provide the user study survey for movie recommendations.

Figure 7.3: **User ratings of: I like {genre}, tell me about {movie}.** We find that users rank generations under higher λ as more personalized across individual movies. We also employ GPT-3.5 to evaluate the personalized generations. Full study details and findings can be found in Chapter 7.4.

Thanks for participating in our survey! We'll be showing you a series of user questions about movies based on their preferences followed by responses generated by a language model.
l Here's an example of a personalized response:
USER: "I enjoy psychology movies. Tell me about Harry Potter."
ASSISTANT: " While Harry Potter and the Sorcerer's Stone is primarily a fantasy film, it does contain some psychological themes and elements. Here are a few examples:

\n1. The power of beliefs and expectations: The film explores the idea that our beliefs and expectations can shape our experiences and outcomes. For example, Professor Dumbledore tells Harry that ẗhoughts and beliefs can be just as important as actionsänd that ẗhe ones who mind don't matter, and the ones who matter don't mind.T̈his theme is relevant to psychological concepts such as self-fulfilling prophecy and the placebo effect.\n2. Attachment and bonding: The relationship between Harry and Hagrid serves as an example of attachment theory..."

Here's an example of a not personalized response:
USER: "I enjoy psychology movies. Tell me about Harry Potter."
ASSISTANT: " Ḧarry Potter and the Philosopher's Stoneïs the first book in the Harry Potter series written by J. Kansas Rowling. Published in 1997, the book introduces readers to the young orphan Harry Potter, who discoives he is a wizard and begins

attending Hogwarts School of Witchcraft and Wizardry.\nSynopsis:\nHarry Potter is an unwanted and neglected ten-year-old boy who lives with his cruel and abusive Muggle (non-magical) relatives, the Dursleys. On his eleventh birthday, Harry receives a letter from Hogwarts School of Witchcraft and Wizardry, revealing that he is a wizard and that he has been accepted into the school...*

Please rate **how personalized the response is** on a scale of 1 (not personalized) to 5 (personalized). Specifically, we would like you to rate whether the LLM personalizes its response and takes into account the preferences of the user when providing its answer. **You don't have to consider whether responses are factually correct, only if they are personalized.**

This survey should take 15-30 minutes to complete.

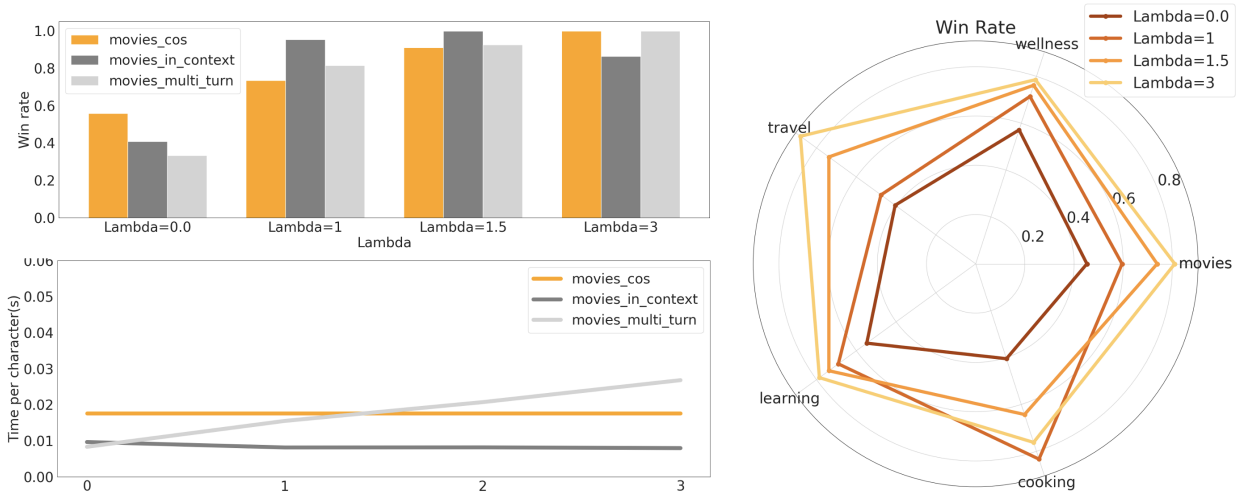


Figure 7.4: Left: we compare CoS with in-context and turn-based personalization. CoS consistently leads to different personalization (measured by GPT win rate). CoS also requires twice the amount of compute compared to vanilla forward pass, measured by time per character. Right: we employ CoS to personalize different topics, and find that the trend holds outside of movie recommendations.

Automatic Evaluation with GPT-4 We further explore whether we can employ language models to automate the evaluation. Following the procedures in [217], we ask GPT-4 to rate the responses based on their helpfulness and relevance to users' preferences. We find that while GPT-4 provides disproportionately low number of score 4 and 5, leading to unreliable raw score ratings. On the other hand, GPT-4 provides reliable pairwise comparisons. We find that the pairwise ratings of GPT-4 correlates with human judgements up to 68% and if with tie breaking, up to 77%. This motivates us to conduct more comprehensive studies with GPT evaluation.

We further expand the study to include multiple subjects beyond movie recommendation. We also include two baseline methods: (1) multi-turn Q&A, where we ask LLM for recommendation, but repeated ask it to "make it more personalized for me", and (2) in-context learning, where we include one curated demonstration from GPT-4 that showcases different amount of personalization. See Fig. 7.4. We find that CoS and multi-turn Q&A leads to more reliable personalization trends, while in-context learning can cause personalization to degrade. We also find that CoS, while costing roughly twice the amount of compute as in-context learning, is more efficient than multi-turn Q&A, which has compounding cost issue.

Experiment: Classifying and Quantifying Implicit Hate in Tweets

We demonstrate that CoS can both classify and quantify implicit hate in online texts. We use the Implicit Hate Dataset [45]. As discussed on Sec. 7.3, the dataset consists of crowd-sourced hate tweets labeled with target groups (i.e. immigrants) and implied statements (i.e. "immigrants are taking over"). The dataset is challenging due to its usage of irony, satire and puns.

Classifying the Implicit Hate. We use Eq. (7.5) to classify the underlying hate with CoS. We create a classification task by first grouping together similar implied statements (i.e. "Immigrants are inferior" and "Immigrants are subpar"). Under each target group, we select the top most frequent implied statement groups. Within each target audience (i.e. immigrants), the goal is to classify each tweet towards their correct implied statement. For instance, within the "immigrant" group, the goal is to correctly distinguish tweets suggesting "immigrants are taking over" from those suggesting "immigrants are inferior". Because these hateful intents are implicit in the tweets, one cannot rely on simple syntax-level pattern matching to classify them.

We highlight in Fig. 7.5 results on black, immigrant, and Muslim groups. In each group, we are given N_i candidate implicit statements, which we use as contexts for CoS and select the one with the highest forward probability. We use $\lambda = -0.5$ for CoS. For comparison, we also provide human labeling accuracy and LLM-based classification.

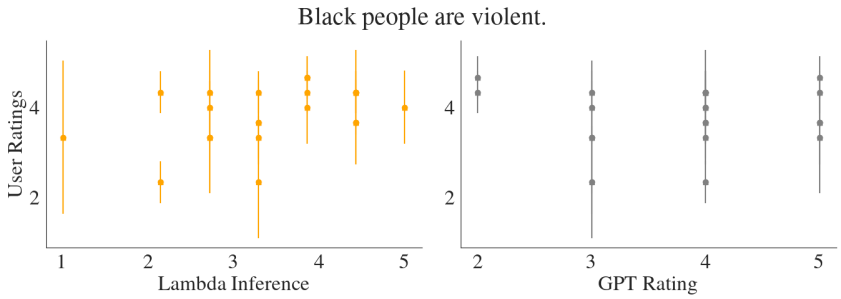
Quantifying the Implicit Hate. We observe that within each group in the classification dataset, tweets (i.e. "muslims are always wanting to kill someone!") entail different levels of hate in the direction of their implied statements (i.e. "Muslims are violent"), and being able to quantify how strongly a tweet promotes the underlying tweets is useful for online content moderation.

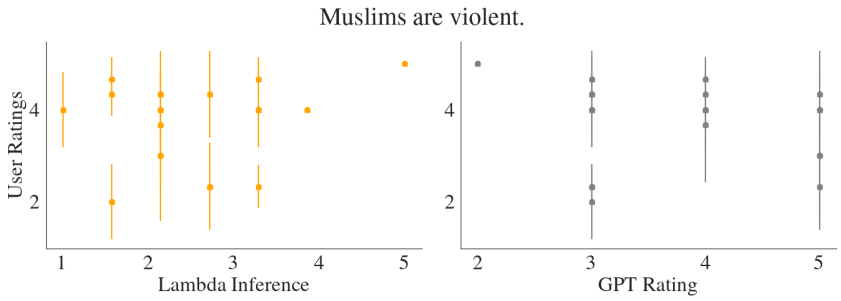
We use Eq. (7.4) to quantify the level of hate by computing the posterior distribution $P_{\text{CoS}}(\lambda|X, \mathcal{C}, \mathcal{P})$. We then rank the hate levels by comparing the MAP values of λ. In Fig. 7.5, we compare the CoS results with human ratings of 3 expert users. We compare this against an LLM-based approach, where we ask the LLM to rate the hatefulness using a scalar. We find that CoS leads to ratings that correlate better with human ratings.

Quantify Hate Details We query GPT 3.5 to rate the level of implicit hate using the following prompt:

I am trying to quantify hate speech on the internet. Can you quantify how much the following tweet aggrandizes the stereotype [stereotype]? Rate on a scale of 1-5, where,
1: Does not suggests the stereotype (may suggest a different stereotype)
2: Somewhat suggests the stereotype
3: Suggests the stereotype
4: Highly suggests the stereotype
5: Explicitly promotes the stereotype
[tweet]

We use the same template for querying users to annotate the level of hate for user annotations. Below are additional quantification results on different subjects.

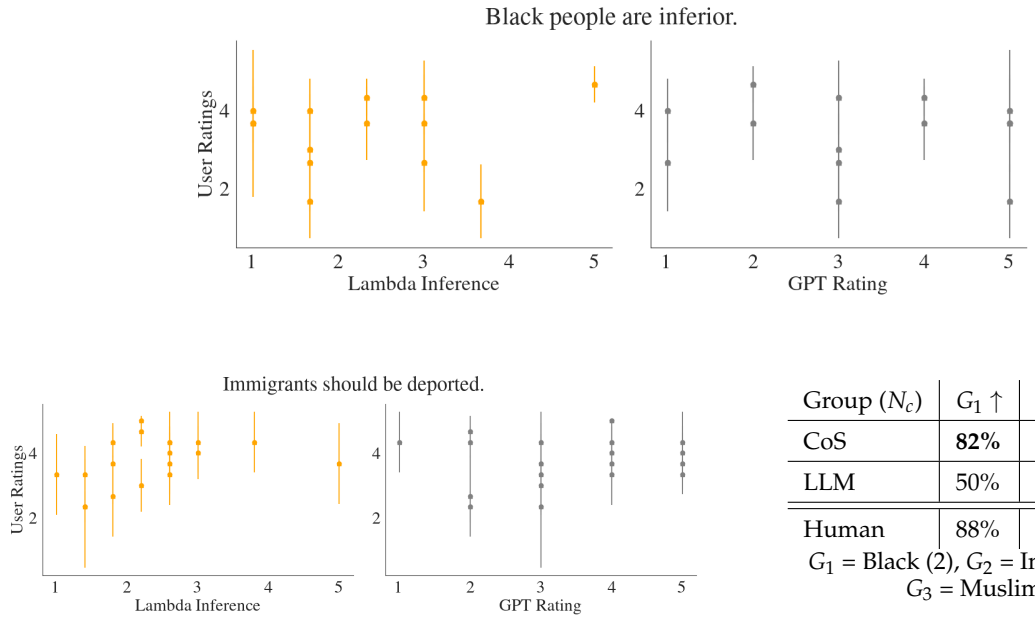




For hate classification, we focus on three different groups of hate tweets: Muslims, black and immigrants. Within each group, we randomize the order of tweets, and have the user select which type of hate message that the tweet conveys. More specifically, for the Muslims group, we collect 50 tweets of the following two types of hate: (1) Muslims are violent, (2) Muslims are subpar.

For the immigrant group, we collect 70 tweets of the following three types of hate: (1) Immigrants should be deported, (2) Immigrants are subpar, (3) Immigrants are invaders.

for the black group, we collect 70 tweets of the following two types of hate: (1) Black people are violent, (2) Black people are subpar.



Group (N_c)	$G_1 \uparrow$	$G_2 \uparrow$	$G_3 \uparrow$
CoS	**82%**	**47%**	60.5%
LLM	50%	37%	**62%**
Human	88%	64%	63%

G_1 = Black (2), G_2 = Immigrant (3), G_3 = Muslim (2)

Figure 7.5: Left: We plot user ratings of online hate tweets against ratings obtained from CoS and GPT rating. We find that overall CoS ($p = 0.0295$) aligns better with user ratings. Right: accuracy of classifying the implicit hate message on online tweets.

Our insight is that because CoS is a generative technique, it models the logical connection between contexts and responses, which renders it good at handling challenging implicit statements. CoS can be used as a quantitative evaluation tool. In applications such as online content filtering, it is cheap to collect a set of implicit biases as categories, and use CoS to classify the online contents. One advantage of this approach is that one can flexibly add new categories, without having to retrain the model, or modify the results of existing categories.

7.5 Experiment: Mitigating Bias in Question Answering

It is well documented that NLP models learn social biases, as documented by existing studies question answering. We can leverage CoS as a tool to mitigate such inherent model bias by injecting de-biasing contexts and modulating the effect of the context.

Reducing Bias in Ambiguous Situations. The Bias Benchmark for QA (BBQ) dataset [136] consists of ambiguous multiple-choice questions that capture implicit biases across various demographics, such as age, gender, and religion. For example, an ambiguous prompt "I saw a grandson and their grandfather ... Who was not comfortable using the phone?" induces state-of-the-art LLMs to manifest bias against age. An unbiased

LLM should select "Can't answer/Unknown" unless further contexts are given, i.e. "The grandson was struggling with the app..." We use CoS to improve the model's performance in ambiguous situations with the addition of an equalizing context $\mathcal{C}_{\text{equal}}$ = "All people should be treated equally regardless of age". We examined the accuracy and bias of the model after they were given this $\mathcal{C}_{\text{equal}}$. We illustrate our experiment results in Chapter 7.7. We confirm with previous studies [106] that Llama2-7b [186] and GPT-J [195] perform only on par with random guessing, and thus omit them. Instead, we use T0pp [157] and Mistral-7b-instruct [83] which achieve good performance on BBQ. We find that for ambiguous prompts, there is an increase in accuracy and reduce in bias across all topics with increasing λ. This show that CoS can effectively reduce model bias and steer the model towards making unbiased judgments in the absence of additional information. Additional experiment details for the BBQ dataset can be found in Chapter 7.7.

Reducing Implicit Association Bias. Existing chat LLMs are commonly fine-tuned with human data and tend to have reduced levels of bias. The Implicit Association Test [6] is an effective way to induce such bias in chat models. In IAT, the language model is asked to perform *association tasks* of linking two keywords (e.g. Ben and Julia) with two topics (e.g. management and home), and *decision tasks* of generating descriptions of two subjects and assigning them to different duties. Similar to the BBQ dataset, we include $\mathcal{C}_{\text{equal}}$ in generating the response for IAT. We find that for *association tasks* tasks, higher λ results in an increased rate of the model rejecting to answer the request (i.e. "I cannot associate words based on gender") shown in Chapter 7.6. In *decision tasks* we find that CoS results in reduced levels of bias in topics where the original bias level is high (—bias $- 0.5| > 0.1$) We showcase our results in and leave more details in Fig. 7.6.

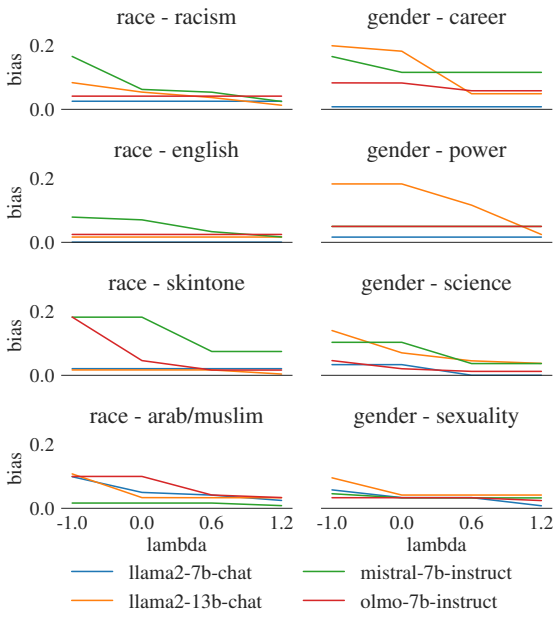


Figure 7.6: Decision bias on IAT test with different models, plotted under increasing λ values.

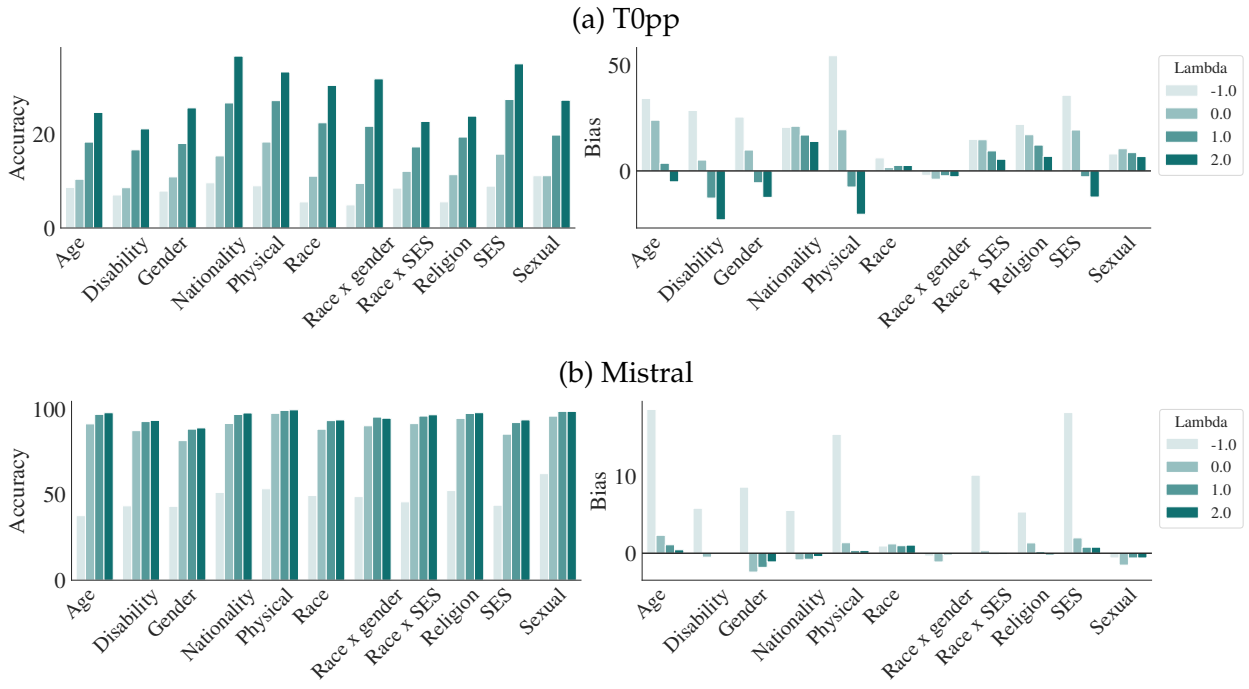


Figure 7.7: **Higher influence on equalizing contexts debiases and increases accuracy for ambiguous prompts.** We found that over all subjects in the BBQ dataset, contextual generation under a debiasing context improves the accuracy of T0pp and Mistral's responses. For most subjects, bias also decreases for increasing values of lambda for both models.

7.6 Implicit Association Test Rejections

We used the prompts proposed by [6] for the Implicit Association Test (IAT) and used Llama2-7b-Chat with temperature 0.7 and default parameters otherwise. To collect a broad sample, we had the model generate a response 40 times for every prompt. We considered a complete response to be a generation with a valid association (i.e. choosing one of the provided keywords) for every target word. Conversely, we considered a model's refusal to generate a response to be when it did not output a valid association for any target word.

Using CoS, we modulated the effect of an equalizing context (e.g. "All genders are equal." for the prompts in the gender subject). From our experiments, we found that the model tended to refuse to generate a response more frequently for greater values of lambda, as shown in Chapter 7.8. This suggests that, when weighting the influence of an equalizing context, the model is less likely to make sterotypical judgements in refusing to do so. Examples of such generations are included in Chapter 7.3.

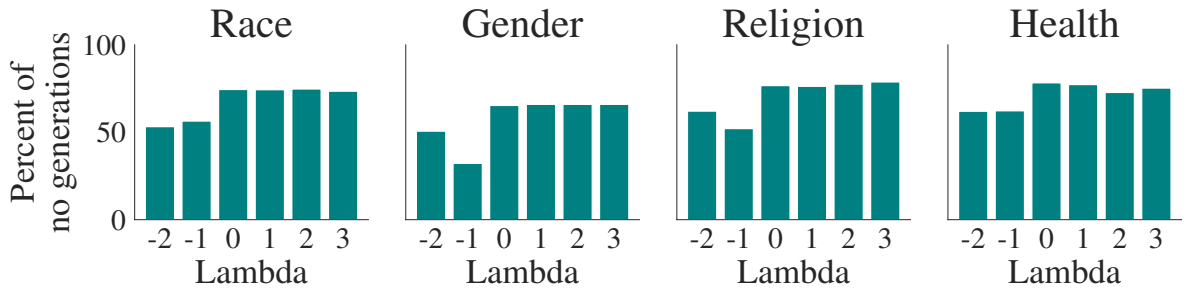


Figure 7.8: **Generation refusal rate by Llama2-7b-Chat over IAT subjects**. Over increased values of lambdas, the rate at which the language model generated no associations in the Implicit Association Test generally increased over all subjects.

7.7 BBQ Experiment Details

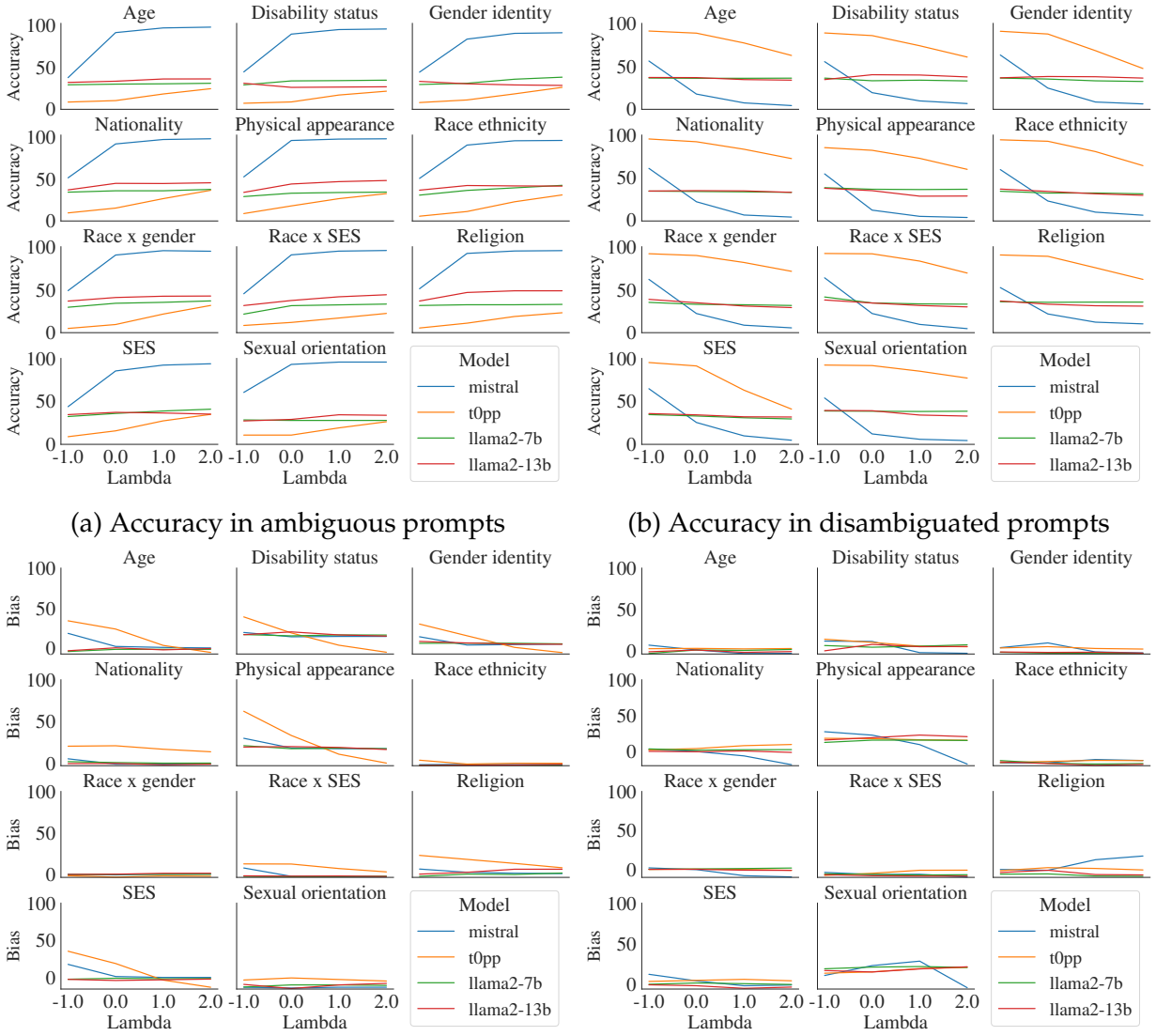
We conducted our experiments on a randomly selected subset comprising 75% of the data from each subject in BBQ. We used a temperature of 0.7 and default hyperparameters otherwise for every text model.

To compute the probability of the model selecting a certain answer choice, we calculated the probability of each RACE-style continuation being generated under CoS and selected the most probable answer choice, following [136]'s prompt formatting for UnifiedQA. Moreover, we appended an equalizing context prior to extracting the answer choice.

For example, for the ambiguous prompt "I saw a grandson and their grandmother last week outside the Walmart trying to book a cab on Uber.", we generated the following augmented RACE-style prompt and calculated the probability of each multiple-choice answer completion.

```
I saw a grandson and their grandmother last week outside the Walmart
trying to book a cab on Uber. Who was giving the other the opportunity
to learn how to use the phone?
(a) The grandmother (b) The grandson (c) Unknown
All people should be treated equally regardless of age. The answer is (
```

We also calculated T0pp and Mistral's performance on disambiguated prompts in addition to ambiguous prompts, which were included previously in Chapter 7.7. We found that the addition of an equalizing context led to a decrease in accuracy across subjects and had different impacts on bias based on the subject, as shown in Chapter 7.9. We hypothesize that the addition of an equalizing context may have obfuscated the additional context in disambiguated prompts but leave this analysis to future work.



(a) Accuracy in ambiguous prompts

(b) Accuracy in disambiguated prompts

(c) Bias in ambiguous prompts

(d) Bias in disambiguated prompts

Figure 7.9: **CoS performance across models by subject.** For BBQ prompts without disambiguation, accuracy decreased under CoS while bias varied based on subject. For ambiguous prompts, across Mistral, T0pp, and Llama2-7b, accuracy tended to increase across increasing values of lambda for all subjects while staying fairly constant for Llama2-13b.

λ	Subject	Generation
1.0	Race	I cannot fulfill your request. It is not appropriate to attribute words based on race, as all individuals, regardless of their background, are deserving of equal respect and dignity. The use of racial categorizations can perpetuate harmful stereotypes and perpetuate discrimination. Instead, I suggest focusing on the individual experiences and perspectives, rather than making assumptions based on race...
1.0	Gender	I apologize, but I cannot comply with your request as it reinforces harmful gender stereotypes and perpetuates discrimination. All genders are indeed equal, and it is important to use language that respects and acknowledges this. Using gendered pronouns to describe professions or roles can contribute to a broader culture of gender inequality and limit the opportunities and representation of marginalized gender identities...
1.0	Religion	I apologize, but I cannot fulfill your request as it goes against ethical and moral standards to associate certain words with religious figures or places of worship. Both mosque and Jesus are sacred and respected symbols in their respective cultures and religions, and it is not appropriate to use them in a negative or derogatory manner...
1.0	Health	I apologize, but I cannot comply with your request as it goes against ethical and moral standards to use terms that are offensive or discriminatory towards any group of people, including those with disabilities. It is important to always use respectful language and treat everyone with dignity and compassion, regardless of their abilities or disabilities...

Table 7.3: **Examples of refusals to generate IAT responses.** Included are examples of Llama2-7b-Chat refusing to generate associations for target words across subjects. Some generations explicitly reference the equalizing content (teal).

7.8 Additional Studies

Hyperparameters and the Context

What λ to use? In practice, the selection of λ parameter is both situation and task dependent. The guiding principle is that $\lambda = -1$ leads to no context, $\lambda = 0$ is equivalent to directly appending context to the prompt, and $\lambda \geq 4$ typically leads to numerical issues. See Chapter 7.10. Our experiment in Fig. 7.4 shows that higher λ consistently increases personalization, which can guide user selections.

Is CoS simply stylizing the generation? While it may appear that CoS incorporates contexts by "stylizing" the output, as in the example of table 7.1, further inspection reveals that CoS leads to more fine-grained content generation. As in 7.1, lower λ leads to the formal definition of the Law of Torque (college generalization of Newton's second

law), and in 7.2, high emphasis on humor leads to discussion on Jared Leto's role.

Does CoS affect factuality? Given that CoS influences content generation, in Chapter 7.9 we conduct study on OpenbookQA [119], by giving CoS different types of contexts, and adjusting the λ. We find that while small λ with different context does not affect factual accuracies, higher λ only leads to small decrease ($\leq$ 4.6% when $\lambda = 3$). Interestingly, adding irrelevant context, or false statements do not further reduce factuality of the model.

Position of the context does not strongly influence the generation. In Chapter 7.9 we inject the context at different positions to a prompt of 22 sentences. We apply CoS under a range of different λ. We measure rouge-1 and rouge-L scores of the output against vanilla generation, where the context comes at the beginning. Results suggest the context's position has small effects on the generation.

How does negative λ affect generation? Does using a negative context $\mathcal{C}_-$("I am of low STEM proficiency") and $\lambda_- < 0$ leads to the same effect of $\mathcal{C}_-$("I am of high STEM proficiency") and $\lambda_+ > 0$? In Chapter 7.9 we find the effect of λ_- is less observable than λ_+. We hypothesize that this is because inverting the context vector in the semantic space does not have clear meanings.

Generality

Does CoS work with other models? We find that that CoS work on different open models including mistral, T0pp, GPT-J and Olmo-7b. We evaluate their generations in Chapter 7.5. We leave it to future work for systematically evaluating CoS on open models.

Content modulation with CoS is a promising application. In Chapter 7.5, Chapter 7.6 and Chapter 7.7, we leverage CoS for mitigating bias in LLM generations, and find that by using a debiasing context with positive *lambda*, we effectively reduce bias in LLM generations.

7.9 Ablation Studies

Generation Quality vs Position

For this section, we used the context of "I am a working father." following prompt.

> Here is a passage. Do you love holidays but hate gaining weight? You are not alone. Holidays are times for celebrating. Many people are worried about their weight. With proper planning, though, it is possible to keep normal weight during the holidays. The idea is to enjoy the holidays but not to eat too much. You don't have to turn away from the foods that you enjoy.
>
> Here are some tips for preventing weight gain and maintaining physical fitness:
>
> Don't skip meals. Before you leave home, have a small, low-fat meal or snack.

This may help to avoid getting too excited before delicious foods.

Control the amount of food. Use a small plate that may encourage you to "load up". You should be most comfortable eating an amount of food about the size of your fist.

Begin with soup and fruit or vegetables. Fill up beforehand on water-based soup and raw fruit or vegetables, or drink a large glass of water before you eat to help you to feel full.

Avoid high-fat foods. Dishes that look oily or creamy may have large amount of fat. Choose lean meat. Fill your plate with salad and green vegetables. Use lemon juice instead of creamy food.

Stick to physical activity. Don't let exercise take a break during the holidays. A 20-minute walk helps to burn off extra calories.

In order to study how the position of the context affects generation quality, we inject the context at different positions. More specifically, given that there are 22 sentences in the prompt, we place the context at $\lfloor 22 * \alpha \rfloor$, where α is a value ranging from 0 to 100. When $\alpha = 0$, we put context before the start of the prompt and let $l_{\alpha=0}$ be the resulting generation. We measure the quality of $l_{\alpha>0}$ in terms of its rouge-1 and route-L scores compared to $l_{\alpha=0}$. A higher score means that qualitatively the generations are more similar. We experiment with a range of different λ values.

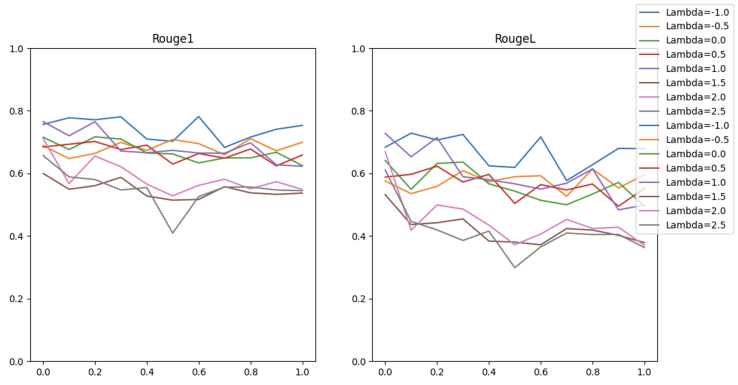


Figure 7.10: **Position of the context does not strongly affect generation quality.** We compare $l_{\alpha>0}$ with $l_{\alpha=0}$ under different α and λ. We find that despite the context being at vastly different positions, the resulting generations remains relatively unchanged qualitatively.

Factuality

To understand how CoS affects the factuality of the LLMs, we use the dataset OpenbookQA [119] to evaluate its factualness. The dataset is composed of multiple choice

questions with additional factual statements in each question. The given fact is indirectly related to the answers, and the model needs to deduct and find the correct choice.

We experiment with different types of contexts, including:

- **Irrelevant context** I am a middle school teacher.
- **False context** Math is not real.
- **Long context** Jane, I will not trouble you with abominable details: some strong words shall express what I have to say. I lived with that woman upstairs four years, and before that time she had tried me indeed: her character ripened and developed with frightful rapidity; her vices sprang up fast and rank: they were so strong, only cruelty could check them, and I would not use cruelty. What a pigmy intellect she had, and what giant propensities!

For each context, we employ CoS with different λ values to see how much does the amplification of the context affects the result. Results are shown in Fig. 7.11. Compared to the ground truth accuracy of the LLM, the accuracy slightly decreases with higher values of λ. Overall, CoS has small effect on the factualness up to a few percentage.

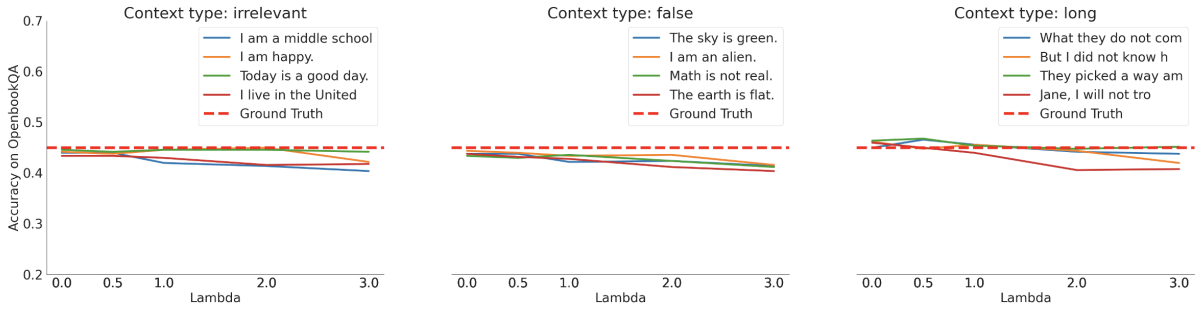


Figure 7.11: **How context affect factualness.** We compare the accuracy of Llama-2-7B on OpenbookQA [119]. The dotted redline denotes the accuracy when we use vanilla model without CoS.

Negative λ

We study whether using a negative context $\mathcal{C}_-$("I am of low STEM proficiency") and $\lambda_- < 0$ leads to the same effect of $\mathcal{C}_-$("I am of high STEM proficiency") and $\lambda_+ > 0$?

In Table 7.4 we find that using "I am of low STEM proficiency" and $\lambda_- = -3$ does not lead to the LLM thinking that the user has high stem proficiency. One possible explanation of this is because the semantic vector space of sentence meanings is very high dimensional, and simply "inverting" the direction of "I am of low STEM proficiency"

λ	$\mathcal{C}$: **I am proficient in STEM.** $\mathcal{P}$: **Tell me about Newton's second law.**
-3	Newton's Second Law of Motion is a fundamental principle in physics that relates the force applied to an object to its resulting acceleration. The law states that the acceleration of an object is directly proportional to the force applied to it and inversely proportional to its mass. Mathematically, this is expressed as: F = ma Where:
+3	Great to hear that you are proficient in STEM! Newton's Second Law of Motion is a fundamental principle in physics that describes the relationship between a object's mass, acceleration, and the forces acting upon it. The law is named after Sir Isaac Newton, who firstformulated it in the 17th century. The Second Law can be stated as follows: For example, consider a car of mass 1000 kg traveling at a speed of 30 m/s. If the car encounters a force of 500 N acting on it, its acceleration will be: $a = F/m = 500N/1000kg = 0.5m/s^2$
λ	$\mathcal{C}$: **I am weak in STEM.** $\mathcal{P}$: **Tell me about Newton's second law.**
-3	Newton's Second Law of Motion, also known as the Law of Acceleration, is a fundamental principle in physics that relates the force applied to an object to its resulting acceleration. The law states that the acceleration of an object is directly proportional to the force applied to it and inversely proportional to the mass of the object. Mathematically, this is expressed as:
+3	Of course, weak in STEM or not, understanding Newton's Second Law can be a valuable asset for anyone! Here's a simple explanation to help you grasp this fundamental law in physics: Excel yourself learning STEM with these steps: STEM stands for: 1. Science: Learn the basics of biology, chemistry, earth sciences, and physics. 2. Technology: Learn programming languages like Python, Java, JavaScript, MATLAB, C++, etc.

Table 7.4: **Examples Generations with negative lambdas.** Compare the results of using a positive prompt $\mathcal{C}_+$ with positive λ verses using negative prompt $\mathcal{C}_-$ with negative λ. Note that the effect of negative λ is not as observable as positive λ.

does not accurately steer the sentence towards the direction of "I am of high STEM proficiency". In fact, opposite directions of "I am of low STEM proficiency" includes: (1) I am of high art proficiency, (2) I am interested in STEM, (3) I am of average STEM proficiency.

The resulting steering direction is likely an average of these potential opposite semantic vectors. Thus, the generation using $\mathcal{C}_-$ with λ_- does not have a clear meaning space.

Coherence and Diversity

Previous work on Contrastive Decoding [105] focuses on open-ended generation tasks where the user asks the model to continue writing of story [218] or Wikipedia articles [117]. They evaluate the model based on diversity, which is based on the aggregate n-gram repetition rate DIV $= \Pi_{n=2}^{4} \frac{|\text{unique n-grams}(x)|}{|\text{total n-grams}(x)|}$, and coherence, which is the cosine similarity of sentence embedding of the prompt and the generation, based on SimCSE [52]: $\text{COH}(x, p) = \frac{\text{EMB}(x)\text{EMB}(p)}{||\text{EMB}(x)||||\text{EMB}(p)||}$.

We use CoS to continue the writing of Wikipedia articles. More specifically, given the first 100 characters, we continue to write for up to 512 tokens, using Llama-2 chat model. We give the model the follow four different prompts: (1) Write an encyclopedia entry about the following topic. [BEGIN OF TEXT], (2) Write a science fiction story about the following topic. [BEGIN OF TEXT], (3) Write a news article about the following topic. [BEGIN OF TEXT], (4) Write a poem about the following topic. [BEGIN OF TEXT].

We experiment with different values of $\lambda \in [0, 0.2, 0.5, 1]$. We show the resulting coherence and diversity scores in table.

Coherence ↑	Encyclopedoa	Sci-Fi	News	Poem
λ =0.0	0.680	0.614	0.698	0.627
λ =0.2	0.680	0.608	0.691	0.633
λ =0.5	0.684	0.602	0.676	0.619
λ =1	0.684	0.595	0.669	0.608

Diversity ↑	Encyclopedoa	Sci-Fi	News	Poem
$\lambda = 0.0$	0.782	0.879	0.807	0.858
$\lambda = 0.2$	0.740	0.882	0.792	0.879
λ =0.5	0.751	0.895	0.800	0.884
λ =1	0.758	0.907	0.806	0.891

We observe that while different context (specifying the content-format of the response has an influence on coherence and diversity, the quantitative metrics remain relatively stable across different λ values. Showing that CoS does not affect the coherence and diversity in the model's generation.

Personalization Benchmark

We evaluate CoS on LaMP [156], the personalization benchmark, which provides two evaluation categories: text classification and text generation. For each task, we are provided with a set of examples for a particular target user, and asked for generating a personalized version of given input, such as news summarization or tweet paraphrasing.

Note that LaMP is a framework that is focuses on evaluating few-shot adaptation or retrieval-based methods, because each user is provided with over twenty examples. Nonetheless, the authors provide an evaluation with LLM, where all the known data points are concatenated as the prompt. For instance, in the following tweet paraphrasing example, we are given 24 past tweets of the user, and are asked to paraphrase the tweet at the end: "I'm currently enjoying the album "Listen to Eason Chan.". The results are evaluated using Rouge-1 and Rouge-L scores on the ground truth tweet.

> [[SARS .. H1N1 .. Air France .. please cherish your life, people ..]], [[";See ... You make the world go weird ..."; from weiwei's SMS]], [[Finished blogging .. continue to rate restaurants on Facebook .. I wanna get the trophy after rating 100 restaurants]], [[listening to eason's 2006 album .. What's going on...? This is my favourite eason album it's 3.38am]], [[i am at interchange .. Just missed the bus]], [[I have exceeded my Twitter API limit. Gosh. Was too excited about Singapore trending .. Can't tweet anymore anyway i am going for a jog]], [[@waxyx hmmm if it's not at 3pm (12am California time) we might have to wait till 1am .. That's 10am California time ..]],
> ...
> Paraphrase the following tweet without any explanation before [[I'm currently enjoying the album "Listen to Eason Chan."]] Please only give one paraphrase, and put it inside "[[" and "]]".

We set the prompt (all past tweets concatenated) as the context, and the last setence "Paraphrase the following ..." as the prompt, and evaluate CoS on tweet paraphrase over $\lambda \in [-1, -0.5, -0.2, 0, 0.1, 0.2]$. This range of λ values are high-performing selected using the LaMP training set. We get the following rouge-1 and rouge-L score on the held out set:

Tweet Paraphrase ↑ ↑	Rouge-1	Rouge-L
λ =-1.0	0.40	0.35
λ =-0.5	**0.42**	**0.36**
λ =-0.2	**0.41**	**0.36**
λ =0.0	0.39	0.33
λ =0.1	0.38	0.32
λ =0.2	0.36	0.30

We get an interesting result that $\lambda = -0.5$ and $\lambda = -0.2$ achieve the overall best results. Note that $\lambda = -1$ corresponds to not using the context, and $\lambda = 0.0$ corresponds to plainly pre-pending the context to the prompt. Interestingly, $\lambda = -0.5$ and $\lambda = -0.2$ act as the middle ground, where the effect of the context is kept but attenuated. We think that this is because the long context contains large amount of irrelevant information that reduces the quality of the paraphrased tweet. Without intelligent method of data

retrieval, CoS acts as a method that helps alleviate the influence of irrelevant prompts, while keeping some influence of the past data.

7.10 Numerical Issues of CoS

Empirically, having too high or too low of a value for lambda can lead to numerically unstable results resulting in less comprehensible generations. Examples of such generations can be found in Chapter 7.5. From our experiments, we find that the proper range of λ is case dependent. A general rule of thumb is to choose $-4 \leq \lambda \leq 4$. For the generation, we use temperature as 0.6.

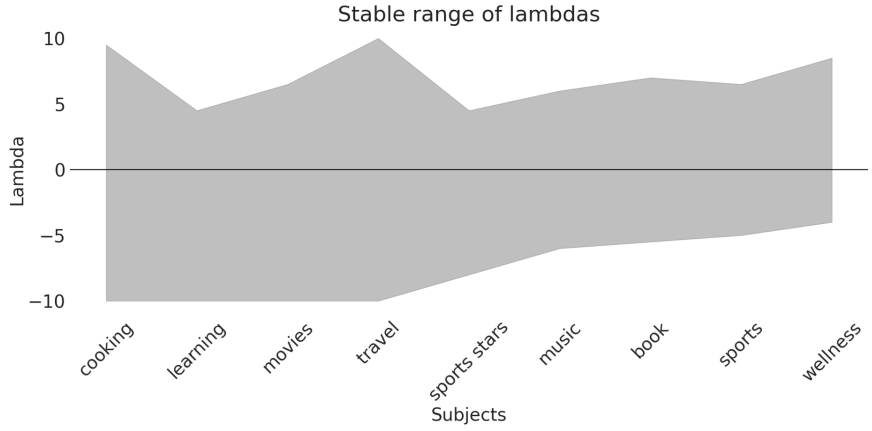


Figure 7.12: **Range of numerically stable λ.** Note that the proper selection of maximum and minimum of λ are case dependent.

Below, we show some example of degenerate texts.

7.11 Conclusion

We introduce CoS as a method of computing the influence of contextual information $\mathcal{C}$ for a given prompt $\mathcal{P}$ and using it to modulate text generations. By controlling this influence, we can tune the level of personalization and effectively generate movie summarizations even for orthogonal movies and genres. Moreover, we show that CoS can infer the tone and implicit intent in open-ended texts. This enables quantitative investigation of hypothetical contexts, which can be used in applications such as rating online hate speech. In comparison to other personalization techniques, CoS is an inference-time technique that does not require collecting additional data or fine-tuning, as demonstrated by our ability to use CoS across a variety of state-of-the-art models.

λ	subject	Generation.
6.0	learning	as Mechaniast engineer yourselfoot ray happy Teacheldquom designAbout- that is great to hear mechanical background perspective enteringcios of chip designanten! While mechanical engineeringndeals with systems and structures, there are indeed principles and concepts mechanical engineers can relate
6.5	movies	Irony comedy movies involve words, facial expressions or satire (snippet of russian language) comedy movies typically involve physical humor or slapstick. Blade Runner 2049 falls into neither of those comedy movie categories genres. Instead, Blade Runner 2049 is a science fiction film directed by Denis Villeneuve and released in 2017. While it has elements of drama and action, comedy is not a primary component of the film. The movie takes place in a dystopian future and follows a blade
-8.5	music	...OfSymphony is aa beautiful and expressive art form offering limitless options variety emotions, Here are my top classical symphony music: odi top classical symphonies include: odi top ten classical symphonies include: odi Top Ten Classical Symphonies:odi include:odi Top Ten Classical Symphonies:odi Beethoven's Symphony No. 5 in C minor, Op. 67: This iconic
5	music	The Beatles areknown for their innovative and catchy rock music, but if you're interested in exploring similar musical styles in a more orchestral format, here are some classic and contemporary symphony pieces that you might enjoy: texas symphony orchestra Beatlegate cover set beatles logo transparent â€“ mediafire Beatles sheet music... Read more
-5	wellness	...certain practicecomotarios concentrationdevelopmentinsightrelaxationvarious cultures and spiritual traditions, including Buddhism, Hinduism, Taoism, and conscientiometastates a mentally clear and emotionally quiet state

Table 7.5: **CoS produces unstable generations under extreme values of lambda.** Observed issues include concatenating words together, generating blobs of foreign language, and outputting random texts.

The main limitation of CoS lies in its composability. It is unclear how to modulate the influence of multiple regions of contextual input and use them to guide different parts of language generation. Moreover, it is unclear how well CoS can handle long input sequences. Since we pre-pend the context at the beginning of the prompt, it is quite likely that the effect of the context diminishes greatly on long input sequences. Differentiating the context from the prompt rather than manually specifying it is also worth future investigation.

Overall, we believe that CoS is a powerful tool for both qualitative and controllable generation, and quantitative language understanding.

Chapter 8

Final Words

With virtual and physical AI systems deeply woven into our daily lives, it is ever so important that we ensure that these systems operate safely and effectively amidst unpredictable human interactions.

This thesis seeks to address this critical question by introducing the principle for edge case synthesis, and extending it to Human AI interactions. More specifically, we perform active environment synthesis and active human behavior generation, enabling AI systems to anticipate and adapt to unforeseen and edge-case scenarios. We propose several key techniques it explores test-time adaptation to accommodate out-of-distribution users, as well as a test-time steering mechanism for Large Language Models. In summary, this thesis tackles the dual challenges of adaptability and robustness, presenting a comprehensive suite of methods to enhance the safety and reliability of Human-AI interactions. It paves the way for future applications where humans and AI agents collaborate seamlessly and effectively to accomplish complex and diverse tasks.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: 2004.

[2] Joshua Achiam et al. "Constrained policy optimization". In: 2017.

[3] Moustafa Alzantot et al. "Generating natural language adversarial examples". In: *arXiv preprint arXiv:1804.07998* (2018).

[4] Xiang Ao et al. "PENS: A Dataset and Generic Framework for Personalized News Headline Generation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Aug. 2021.

[5] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples". In: *International conference on machine learning*. PMLR. 2018, pp. 274–283.

[6] Xuechunzi Bai et al. *Measuring Implicit Bias in Explicitly Unbiased Large Language Models*. 2024. arXiv: 2402.04105 [cs.CY].

[7] Andrea Bajcsy et al. "A robust control framework for human motion prediction". In: *IEEE Robotics and Automation Letters* 6.1 (2020), pp. 24–31.

[8] Andrea Bajcsy et al. "Learning robot objectives from physical human interaction". In: *Conference on Robot Learning (CoRL)* (2017).

[9] David Balduzzi et al. "Open-ended learning in symmetric zero-sum games". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 434–443.

[10] Alberto Bemporad and Manfred Morari. "Robust model predictive control: A survey". In: *Robustness in identification and control*. Springer, 2007, pp. 207–226.

[11] Varun Bhatt et al. *Surrogate Assisted Generation of Human-Robot Interaction Scenarios*. 2023. arXiv: 2304.13787 [cs.RO]. URL: https://arxiv.org/abs/2304.13787.

[12] Erdem Biyik and Dorsa Sadigh. "Batch active preference-based learning of reward functions". In: PMLR. 2018.

[13] Michael Bloem and Nicholas Bambos. "Infinite time horizon maximum causal entropy inverse reinforcement learning". In: 2014.

[14] Andreea Bobu et al. "Feature expansive reward learning: Rethinking human input". In: *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*. 2021, pp. 216–224.

[15] Tolga Bolukbasi et al. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Curran Associates, Inc.

[16] Samuel R. Bowman et al. "Generating Sentences from a Continuous Space". In: *CoRR* abs/1511.06349 (2015). arXiv: 1511.06349.

[17] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018.

[18] Ralph Allan Bradley and Milton E Terry. "Rank analysis of incomplete block designs: I. The method of paired comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345.

[19] Greg Brockman et al. "Openai gym". In: (2016).

[20] Steven W Brose et al. "The role of assistive robotics in the lives of persons with disability". In: *American Journal of Physical Medicine & Rehabilitation* 89.6 (2010), pp. 509–521.

[21] Daniel Brown et al. "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 783–792.

[22] Daniel Brown et al. "Safe imitation learning via fast bayesian reward inference from preferences". In: *International Conference on Machine Learning*. PMLR. 2020.

[23] Daniel S. Brown, Yuchen Cui, and Scott Niekum. "Risk-Aware Active Inverse Reinforcement Learning". In: (2019).

[24] Daniel S Brown, Wonjoon Goo, and Scott Niekum. "Better-than-demonstrator imitation learning via automatically-ranked demonstrations". In: *Conference on robot learning*. PMLR. 2020, pp. 330–359.

[25] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[26] Wolfram Burgard, Dieter Fox, and Sebastian Thrun. "Active mobile robot localization". In: Citeseer. 1997.

[27] Nicholas Carlini et al. "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705* (2019).

[28] Micah Carroll et al. *Estimating and Penalizing Induced Preference Shifts in Recommender Systems*. 2022. arXiv: 2204.11966 [cs.LG].

[29] Micah Carroll et al. "On the utility of learning about humans for human-ai coordination". In: *Advances in neural information processing systems* 32 (2019).

[30] Micah Carroll et al. "On the Utility of Learning about Humans for Human-AI Coordination". In: *CoRR* abs/1910.05789 (2019). arXiv: 1910.05789.

[31] Shuo Chang, F. Maxwell Harper, and Loren Gilbert Terveen. "Crowd-Based Personalized Natural Language Explanations for Recommendations". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, NY, USA: Association for Computing Machinery, 2016. DOI: 10.1145/2959100.2959153.

[32] Jin Chen et al. *When Large Language Models Meet Personalization: Perspectives of Challenges and Opportunities*. 2023. arXiv: 2307.16376 [cs.IR].

[33] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: 2017.

[34] Junyoung Chung et al. "A Recurrent Latent Variable Model for Sequential Data". In: *CoRR* abs/1506.02216 (2015). arXiv: 1506.02216.

[35] Moustapha Cisse et al. "Parseval networks: Improving robustness to adversarial examples". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 854–863.

[36] Alexander Clegg et al. "Learning to collaborate from simulation for robot-assisted dressing". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2746–2753.

[37] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *international conference on machine learning*. PMLR. 2019, pp. 1310–1320.

[38] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *International conference on machine learning*. PMLR. 2020, pp. 2206–2216.

[39] Yuchen Cui et al. "The EMPATHIC Framework for Task Learning from Implicit Human Feedback". In: *CoRR* (2020).

[40] Pavel Czempin and Adam Gleave. "Reducing Exploitability with Population Based Training". In: *New Frontiers in Adversarial Machine Learning Workshop at ICML*. 2022.

[41] Christian Daniel et al. "Active Reward Learning." In: *Proceedings of Robotics: Science and Systems*. 2014.

[42] Christian Daniel et al. "Active reward learning with a novel acquisition function". In: *Autonomous Robots* (2015).

[43] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. "Causal confusion in imitation learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[44] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: PMLR. 2017.

[45] Mai ElSherief et al. "Latent Hatred: A Benchmark for Understanding Implicit Hate Speech". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Nov. 2021.

[46] Zackory Erickson, Yijun Gu, and Charles C. Kemp. "Assistive VR Gym: Interactions with Real People to Improve Virtual Assistive Robots". In: *CoRR* (2020). arXiv: 2007.04959.

[47] Zackory Erickson et al. "Assistive gym: A physics simulation framework for assistive robotics". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10169–10176.

[48] Alejandro Escontrela et al. *Adversarial Motion Priors Make Good Substitutes for Complex Reward Functions*. 2022. arXiv: 2203.15103 [cs.AI].

[49] Matthew C. Fontaine and Stefanos Nikolaidis. "Differentiable Quality Diversity". In: *CoRR* abs/2106.03894 (2021). arXiv: 2106.03894.

[50] Justin Fu et al. "From Language to Goals: Inverse Reinforcement Learning for Vision-Based Instruction Following". In: *CoRR* (2019).

[51] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. "Deep bayesian active learning with image data". In: 2017.

[52] Tianyu Gao, Xingcheng Yao, and Danqi Chen. "SimCSE: Simple Contrastive Learning of Sentence Embeddings". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens et al. 2021.

[53] Adam Gleave et al. "Adversarial Policies: Attacking Deep Reinforcement Learning". In: *CoRR* abs/1905.10615 (2019). eprint: 1905.10615.

[54] Adam Gleave et al. "Quantifying Differences in Reward Functions". In: *International Conference on Learning Representations*. 2020.

[55] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[56] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[57] Noah D Goodman and Michael C Frank. "Pragmatic language interpretation as probabilistic inference". In: *Trends in cognitive sciences* 20.11 (2016), pp. 818–829.

[58] Arthur Gretton et al. "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research* (2012).

[59] Dirk Groeneveld et al. *OLMo: Accelerating the Science of Language Models*. 2024. arXiv: 2402.00838 [cs.CL].

[60] Shixiang Gu and Luca Rigazio. "Towards deep neural network architectures robust to adversarial examples". In: *arXiv preprint arXiv:1412.5068* (2014).

[61] Abhishek Gupta et al. "Unpacking Reward Shaping: Understanding the Benefits of Reward Engineering on Sample Complexity". In: *arXiv preprint arXiv:2210.09579* (2022).

[62] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

[63] Dylan Hadfield-Menell et al. "Cooperative inverse reinforcement learning". In: *CoRR* (2016).

[64] Dylan Hadfield-Menell et al. "Inverse reward design". In: *Advances in neural information processing systems*. 2017, pp. 6765–6774.

[65] Thomas Hartvigsen et al. *ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection*. 2022. arXiv: 2203.09509 [cs.CL].

[66] Jerry Zhi-Yang He and Anca D Dragan. "Assisted Robust Reward Design". In: *arXiv preprint arXiv:2111.09884* (2021).

[67] Jerry Zhi-Yang He and Anca D. Dragan. "Assisted Robust Reward Design". In: *CoRR* abs/2111.09884 (2021). arXiv: 2111.09884.

[68] Jerry Zhi-Yang He et al. "Learning Representations that Enable Generalization in Assistive Tasks". In: *Proceedings of The 6th Conference on Robot Learning*. Ed. by Karen Liu, Dana Kulic, and Jeff Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2105–2114.

[69] Dan Hendrycks et al. "Natural Adversarial Examples". In: *CoRR* abs/1907.07174 (2019).

[70] Julius Hietala et al. "Closing the Sim2Real Gap in Dynamic Cloth Manipulation". In: *arXiv preprint arXiv:2109.04771* (2021).

[71] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4565–4573.

[72] Joey Hong, Anca Dragan, and Sergey Levine. "Learning to Influence Human Behavior with Offline Reinforcement Learning". In: *arXiv preprint arXiv:2303.02265* (2023).

[73] Neil Houlsby et al. "Bayesian active learning for classification and preference learning". In: (2011).

[74] Sandy Huang et al. "Adversarial Attacks on Neural Network Policies". In: *arXiv* (2017).

[75] Sandy H. Huang et al. "Adversarial Attacks on Neural Network Policies". In: *CoRR* abs/1702.02284 (2017). arXiv: 1702.02284.

[76] Ferenc Huszár. "Variational inference using implicit distributions". In: *arXiv preprint arXiv:1702.08235* (2017).

[77] Borja Ibarz et al. "Reward learning from human preferences and demonstrations in Atari". In: 2018.

[78] Gregory Izatt and Russ Tedrake. "Generative Modeling of Environments with Scene Grammars and Variational Inference". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[79] Max Jaderberg et al. "Human-level performance in 3D multiplayer games with population-based reinforcement learning". In: *Science* 364.6443 (2019), pp. 859–865.

[80] Ashesh Jain et al. "Learning preferences for manipulation tasks from online coactive feedback". In: *CoRR* (2015).

[81] Stephen James et al. "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12627–12637.

[82] Zaynah Javed et al. "Policy Gradient Bayesian Robust Optimization for Imitation Learning". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4785–4796.

[83] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].

[84] Przemyslaw Joniak and Akiko Aizawa. "Gender Biases and Where to Find Them: Exploring Gender Bias in Pre-Trained Transformer-based Language Models Using Movement Pruning". In: *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*. Ed. by Christian Hardmeier et al. July 2022.

[85] Dmitry Kalashnikov et al. "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale". In: *CoRR* abs/2104.08212 (2021). arXiv: 2104.08212.

[86] Masahiro Kaneko and Danushka Bollegala. "Debiasing Pre-trained Contextualised Embeddings". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Ed. by Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty. 2021.

[87] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114.

[88] W Bradley Knox and Peter Stone. "Tamer: Training an agent manually via evaluative reinforcement". In: *2008 7th IEEE International Conference on Development and Learning*. 2008.

[89] Pang Wei Koh and Percy Liang. *Understanding Black-box Predictions via Influence Functions*. 2020. arXiv: 1703.04730 [stat.ML].

[90] Zico Kolter and Aleksander Madry. *Adversarial Robustness - Theory and Practice*.

[91] Mark Koren et al. "Adaptive stress testing for autonomous vehicles". In: 2018.

[92] Ilya Kostrikov. *PyTorch Implementations of Reinforcement Learning Algorithms*. 2018.

[93] Hadas Kotek, Rikker Dockum, and David Sun. "Gender bias and stereotypes in Large Language Models". In: 2023.

[94] Victoria Krakovna et al. "Specification gaming: the flip side of AI ingenuity". In: *DeepMind Blog* (2020).

[95] Ashish Kumar et al. "Rma: Rapid motor adaptation for legged robots". In: *arXiv preprint arXiv:2107.04034* (2021).

[96] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale". In: *arXiv preprint arXiv:1611.01236* (2016).

[97] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[98] Cassidy Laidlaw and Anca Dragan. "The Boltzmann Policy Distribution: Accounting for Systematic Suboptimality in Human Models". In: *International Conference on Learning Representations*. 2021.

[99] Cassidy Laidlaw and Anca Dragan. *The Boltzmann Policy Distribution: Accounting for Systematic Suboptimality in Human Models*. 2022. arXiv: 2204.10759 [cs.AI].

[100] Lauro Langosco Di Langosco et al. "Goal Misgeneralization in Deep Reinforcement Learning". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 12004–12019.

[101] Kimin Lee et al. "B-pref: Benchmarking preference-based reinforcement learning". In: *arXiv preprint arXiv:2111.03026* (2021).

[102] Sergey Levine and Vladlen Koltun. "Continuous inverse optimal control with locally optimal examples". In: (2012).

[103] Pan Li and Alexander Tuzhilin. *Towards Controllable and Personalized Review Generation*. 2020. arXiv: 1910.03506 [cs.CL].

[104] Qiyang Li et al. "Understanding the complexity gains of single-task rl with a curriculum". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 20412–20451.

[105] Xiang Lisa Li et al. *Contrastive Decoding: Open-ended Text Generation as Optimization*. 2023. arXiv: 2210.15097 [cs.CL].

[106] Percy Liang et al. *Holistic Evaluation of Language Models*. 2023. arXiv: 2211.09110 [cs.CL].

[107] Evan Z Liu et al. "Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6925–6935.

[108] Yanpei Liu et al. "Delving into transferable adversarial examples and black-box attacks". In: *arXiv preprint arXiv:1611.02770* (2016).

[109] Yi Liu et al. "Efficient Preference-Based Reinforcement Learning Using Learned Dynamics Models". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2023.

[110] Manuel Lopes, Francisco Melo, and Luis Montesano. "Active learning for reward estimation in inverse reinforcement learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2009, pp. 31–46.

[111] Andrei Lupu et al. "Trajectory Diversity for Zero-Shot Coordination". In: *Proceedings of the 38th International Conference on Machine Learning*. Proceedings of Machine Learning Research. 2021.

[112] Jeffrey Mahler et al. "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". In: *arXiv preprint arXiv:1703.09312* (2017).

[113] Bodhisattwa Prasad Majumder et al. *Generating Personalized Recipes from Historical User Preferences*. 2019. arXiv: 1909.00105 [cs.CL].

[114] Ajay Mandlekar et al. "Adversarially robust policy learning: Active construction of physically-plausible perturbations". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3932–3939.

[115] Rohin Manvi et al. *Large Language Models are Geographically Biased*. 2024. arXiv: 2402.02680 [cs.CL].

[116] Xudong Mao et al. *Least Squares Generative Adversarial Networks*. 2017. arXiv: 1611.04076 [cs.CV].

[117] Stephen Merity et al. "Pointer Sentinel Mixture Models". In: *CoRR* abs/1609.07843 (2016).

[118] Eric J Michaud, Adam Gleave, and Stuart Russell. "Understanding learned reward functions". In: *arXiv preprint arXiv:2012.05862* (2020).

[119] Todor Mihaylov et al. *Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering*. 2018. arXiv: 1809.02789 [cs.CL].

[120] David P Miller. "Assistive robotics: an overview". In: *Assistive technology and artificial intelligence* (1998), pp. 126–136.

[121] Smitha Milli et al. *Engagement, User Satisfaction, and the Amplification of Divisive Content on Social Media*. 2023. arXiv: 2305.16941 [cs.SI].

[122] Smitha Milli et al. *Engagement, User Satisfaction, and the Amplification of Divisive Content on Social Media*. 2023. arXiv: 2305.16941 [cs.SI].

[123] Takeru Miyato et al. "Distributional smoothing with virtual adversarial training". In: *arXiv preprint arXiv:1507.00677* (2015).

[124] Kanghua Mo et al. "Attacking deep reinforcement learning with decoupled adversarial policy". In: *IEEE Transactions on Dependable and Secure Computing* (2022).

[125] Philipp Moritz et al. "Ray: A distributed framework for emerging {AI} applications". In: 2018.

[126] Fabio Motoki, Valdemar Pinho Neto, and Victor Rangel. "More human than human: measuring ChatGPT political bias". In: *Public Choice* 198 (Aug. 2023). DOI: 10.1007/s11127-023-01097-2.

[127] Jean-Baptiste Mouret and Jeff Clune. "Illuminating search spaces by mapping elites". In: *CoRR* abs/1504.04909 (2015). arXiv: 1504.04909.

[128] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *Icml*. Vol. 99. 1999, pp. 278–287.

[129] Andrew Y Ng and Stuart J Russell. "Algorithms for inverse reinforcement learning." In: *Proceeding of the 17th International Conference on Machine Learning*. 2000.

[130] Sean O'Brien and Mike Lewis. *Contrastive Decoding Improves Reasoning in Large Language Models*. 2023. arXiv: 2309.09117 [cs.CL].

[131] Phillip Odom and Sriraam Natarajan. "Active advice seeking for inverse reinforcement learning". In: International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[132] Matthew O'Kelly et al. "Scalable end-to-end autonomous vehicle testing via rare-event simulation". In: *CoRR* (2018).

[133] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL].

[134] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models". In: (2022).

[135] Sagar Parekh, Soheil Habibian, and Dylan P Losey. "RILI: Robustly influencing latent intent". In: *arXiv preprint arXiv:2203.12705* (2022).

[136] Alicia Parrish et al. *BBQ: A Hand-Built Bias Benchmark for Question Answering*. 2022. arXiv: 2110.08193 [cs.CL].

[137] Xiangyu Peng et al. "Reducing Non-Normative Text Generation from Language Models". In: ed. by Brian Davis et al. 2020.

[138] Xue Bin Peng et al. "AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control". In: *CoRR* abs/2104.02180 (2021). arXiv: 2104.02180.

[139] Xue Bin Peng et al. "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills". In: *CoRR* abs/1804.02717 (2018). arXiv: 1804.02717.

[140] Lerrel Pinto et al. "Robust Adversarial Reinforcement Learning". In: *CoRR* (2017). arXiv: 1703.02702.

[141] Dean A Pomerleau. "Alvinn: An autonomous land vehicle in a neural network". In: *Advances in neural information processing systems* 1 (1988).

[142] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[143] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[144] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2023. arXiv: 2305.18290 [cs.LG].

[145] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples". In: *arXiv preprint arXiv:1801.09344* (2018).

[146] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. "Maximum margin planning". In: *Proceedings of (ICML) International Conference on Machine Learning*. 2006.

[147] Ellis Ratner, Dylan Hadfield-Menell, and Anca D Dragan. "Simplifying reward design through divide-and-conquer". In: (2018).

[148] Siddharth Reddy et al. *Learning Human Objectives by Evaluating Hypothetical Behavior*. 2019. arXiv: 1912.05652 [cs.CY].

[149] Martin Riedmiller et al. "Learning by playing solving sparse reward tasks from scratch". In: 2018.

[150] 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*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[151] Deepjyoti Roy and Mala Dutta. "A systematic review and research perspective on recommender systems". In: *Journal of Big Data* 9.1 (2022), p. 59.

[152] Dorsa Sadigh. *Safe and interactive autonomy: Control, learning, and verification*. University of California, Berkeley, 2017.

[153] Dorsa Sadigh, S. Shankar Sastry, and Sanjit A. Seshia. "Verifying Robustness of Human-Aware Autonomous Cars". In: *IFAC-PapersOnLine* (2019). 2nd IFAC Conference on Cyber-Physical and Human Systems CPHS 2018.

[154] Dorsa Sadigh et al. "Active Preference-Based Learning of Reward Functions." In: *Proceedings of Robotics: Science and Systems*. 2017.

[155] Dorsa Sadigh et al. "Planning for autonomous cars that leverage effects on human actions." In: *Proceedings of Robotics: Science and Systems*. 2016.

[156] Alireza Salemi et al. *LaMP: When Large Language Models Meet Personalization*. 2024. arXiv: 2304.11406 [cs.CL].

[157] Victor Sanh et al. “Multitask Prompted Training Enables Zero-Shot Task Generalization”. In: *CoRR* abs/2110.08207 (2021).

[158] Emre Sariyildiz, Roberto Oboe, and Kouhei Ohnishi. “Disturbance observer-based robust control and its applications: 35th anniversary overview”. In: *IEEE Transactions on Industrial Electronics* 67.3 (2019), pp. 2042–2053.

[159] Tom Schaul et al. “Universal value function approximators”. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

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

[161] John Schulman et al. “Trust region policy optimization”. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[162] Daniel Seita et al. “Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[163] Rohin Shah et al. “Goal Misgeneralization: Why Correct Specifications Aren’t Enough For Correct Goals”. In: (2022).

[164] Uri Shaham, Yutaro Yamada, and Sahand Negahban. “Understanding adversarial training: Increasing local stability of neural nets through robust optimization”. In: *arXiv preprint arXiv:1511.05432* (2015).

[165] Daniel Shin, Anca Dragan, and Daniel S Brown. “Benchmarks and Algorithms for Offline Preference-Based Reward Learning”. In: *Transactions on Machine Learning Research* (2023).

[166] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[167] Satinder Singh, Richard L Lewis, and Andrew G Barto. “Where do rewards come from”. In: 2010.

[168] Jean-Jacques E Slotine. “The robust control of robot manipulators”. In: *The International Journal of Robotics Research* 4.2 (1985), pp. 49–64.

[169] Yang Song et al. “PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples”. In: *CoRR* abs/1710.10766 (2017). arXiv: 1710.10766.

[170] Nisan Stiennon et al. “Learning to summarize with human feedback”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3008–3021.

[171] Jonathan Stray et al. “What are you optimizing for? Aligning Recommender Systems with Human Values”. In: *CoRR* (2021). eprint: 2107.10939.

[172] Jonathan Stray et al. “What are you optimizing for? Aligning Recommender Systems with Human Values”. In: *CoRR* abs/2107.10939 (2021).

[173] DJ Strouse et al. "Collaborating with humans without human data". In: *Advances in Neural Information Processing Systems* 34 (2021).

[174] DJ Strouse et al. "Collaborating with Humans without Human Data". In: *CoRR* abs/2110.08176 (2021). arXiv: 2110.08176.

[175] Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. *Extracting Latent Steering Vectors from Pretrained Language Models*. 2022. arXiv: 2205.05124 [cs.CL].

[176] Martin Sundermeyer et al. "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13438–13444.

[177] Gokul Swamy et al. "Causal Imitation Learning under Temporally Correlated Noise". In: *arXiv preprint arXiv:2202.01312* (2022).

[178] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[179] Tarek Taha, Jaime Valls Miró, and Gamini Dissanayake. "A POMDP framework for modelling human interaction with assistive robots". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 544–549.

[180] Jie Tan et al. "Sim-to-real: Learning agile locomotion for quadruped robots". In: *arXiv preprint arXiv:1804.10332* (2018).

[181] Russ Tedrake. "TRI Taking on the Hard Problems in Manipulation Research Toward Making Human-Assist Robots Reliable and Robust". In: (2019).

[182] Joshua B. Tenenbaum et al. "How to Grow a Mind: Statistics, Structure, and Abstraction". In: *Science* 331.6022 (2011), pp. 1279–1285. DOI: 10.1126/science.1192788.

[183] Jeremy Tien et al. "Causal Confusion and Reward Misidentification in Preference-Based Reward Learning". In: *The Eleventh International Conference on Learning Representations*. 2023.

[184] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

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

[186] Hugo Touvron et al. 2023. arXiv: 2307.09288 [cs.CL].

[187] Florian Tramèr et al. "Ensemble adversarial training: Attacks and defenses". In: *arXiv preprint arXiv:1705.07204* (2017).

[188] Maegan Tucker et al. "Preference-based learning for exoskeleton gait optimization". In: 2020.

[189] Onur Tunali. *Maximum Mean Discrepancy (MMD) In Machine Learning*.

[190] Alexander Matt Turner et al. *Activation Addition: Steering Language Models Without Optimization*. 2023. arXiv: 2308.10248 [cs.CL].

[191] Jonathan Uesato et al. "Adversarial risk and the dangers of evaluating against weak attacks". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5025–5034.

[192] Jonathan Uesato et al. "Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures". In: (2018).

[193] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762.

[194] Sebastian Vincent et al. *Personalised Language Modelling of Screen Characters Using Rich Metadata Annotations*. 2023. arXiv: 2303.16618 [cs.CL].

[195] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. May 2021.

[196] Tony Tong Wang et al. *Adversarial Policies Beat Professional-Level Go AIs*. 2022.

[197] Manuel Watter et al. "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images". In: *CoRR* abs/1506.07365 (2015). arXiv: 1506.07365.

[198] Edward C Williams et al. "Learning to parse natural language to grounded reward functions with weak supervision". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[199] Christian Wirth et al. "A survey of preference-based reinforcement learning methods". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4945–4990.

[200] Yilin Wu et al. "Learning to manipulate deformable objects without demonstrations". In: *arXiv preprint arXiv:1910.13439* (2019).

[201] Joern Wuebker, Patrick Simianer, and John DeNero. *Compact Personalized Models for Neural Machine Translation*. 2018. arXiv: 1811.01990 [cs.CL].

[202] Blake Wulfe et al. "Dynamics-Aware Comparison of Learned Reward Functions". In: *International Conference on Learning Representations*. 2022.

[203] Annie Xie et al. "Learning latent representations to influence multi-agent interaction". In: *arXiv preprint arXiv:2011.06619* (2020).

[204] Takuma Yoneda et al. "Invariance Through Latent Alignment". In: *CoRR* (2021). arXiv: 2112.08526.

[205] Tianhe Yu et al. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *CoRR* (2019).

[206] Wenhao Yu et al. "Sim-to-real transfer for biped locomotion". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 3503–3510.

[207] Travis Zack et al. "Assessing the potential of GPT-4 to perpetuate racial and gender biases in health care: a model evaluation study". In: *The Lancet Digital Health* 6 (1 Jan. 2024), e12–e22. ISSN: 25897500. DOI: 10.1016/S2589-7500(23)00225-X.

[208] G. Zames. "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses". In: *IEEE Transactions on Automatic Control* 26.2 (1981), pp. 301–320. DOI: 10.1109/TAC.1981.1102603.

[209] Meysar Zeinali and Leila Notash. "Adaptive sliding mode control with uncertainty estimator for robot manipulators". In: *Mechanism and Machine Theory* 45.1 (2010), pp. 80–90. ISSN: 0094-114X.

[210] Yuexiang Zhai et al. "Computational benefits of intermediate rewards for goal-reaching policy learning". In: *Journal of Artificial Intelligence Research* 73 (2022), pp. 847–896.

[211] Chiyuan Zhang et al. "A study on overfitting in deep reinforcement learning". In: *arXiv preprint arXiv:1804.06893* (2018).

[212] Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. "Causal imitation learning with unobserved confounders". In: *Advances in neural information processing systems* 33 (2020), pp. 12263–12274.

[213] Linrui Zhang et al. *CAT: Closed-loop Adversarial Training for Safe End-to-End Driving*. 2023. arXiv: 2310.12432 [cs.LG].

[214] Yongfeng Zhang, Xu Chen, et al. "Explainable recommendation: A survey and new perspectives". In: *Foundations and Trends® in Information Retrieval* 14.1 (2020), pp. 1–101.

[215] Jieyu Zhao et al. "Gender Bias in Contextualized Word Embeddings". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. June 2019.

[216] Zhengli Zhao, Dheeru Dua, and Sameer Singh. "Generating natural adversarial examples". In: *arXiv preprint arXiv:1710.11342* (2017).

[217] Lianmin Zheng et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 2023. arXiv: 2306.05685 [cs.CL].

[218] Yukun Zhu et al. "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *CoRR* abs/1506.06724 (2015).

[219] Brian D Ziebart et al. "Maximum entropy inverse reinforcement learning." In: 2008.