

ABSTRACT

Title of Dissertation: **MULTI-AGENT AUTONOMOUS DECISION MAKING
IN ARTIFICIAL INTELLIGENCE**

Saptarashmi Bandyopadhyay
Doctor of Philosophy, 2025

Dissertation Directed by: Professor John P. Dickerson
Department of Computer Science
Professor Thomas Goldstein
Department of Computer Science

Multi-Agent Autonomous Decision Making, especially Multi-Agent Reinforcement Learning (MARL), is an emerging area of Artificial Intelligence (AI) where autonomous agents interact with each other, fostering competition and/or cooperation. These AI agents can be useful to solve real world problems like Augmented Reality, Recommender Systems, Supply Chain Orchestration, Climate Conservation, Self-Driving Cars, Sports, Interdiction Games and Real-time Guidance of Cooking, Education, Manufacturing and Robotic Tasks. Challenges of AI Agents include efficiently scaling to multiple agents, solving coordination problems and understanding agentic behavior.

My Ph.D. thesis has the objective to develop and deploy efficient Multi-Agent AI algorithms, for real-world decision making problems. To begin with, a Multi-Agent approach can be used to model the Human-AI Alignment problem, a major challenge to rapidly deploy AI

models. Mis-alignment challenges exist in current AI models like ChatGPT which face serious challenges to plan or reason like performing a 4-digit multiplication of two integers. Concepts from MARL like ad-hoc coordination help humans and autonomous agents to communicate the goals of each planned step correctly and explicitly reason about the strategies humans may utilize when attempting to shape the behavior of AI model agents. These kinds of communications are not often efficient or robust with increasing scale of AI Agents and need theoretical results for efficient Agentic behavior. I have provided formal guarantees for successful and reliable cooperation of AI agents with populations of socially intelligent agents, defining agentic behavior with game-theoretic notions of consistency and compatibility. AI agents cooperate in the above settings with populations of socially intelligent agents that are individually rational while also reliably coordinating with other group members in a general-sum Bayesian game. The AI agents face challenges generalizing from previous interactions that can help them to cooperate with a new partner drawn from such populations. It is theoretically shown that just these assumptions are insufficient to select an AI agent's strategy that achieves zero-shot coordination with any member in the socially intelligent population, which can be addressed by a proven upper bound on the sample complexity to learn a successful cooperation strategy, based on observing interaction among members of the target population. Lower bounds are derived to show when the Multi-Agent cooperation setting is needed with respect to the populations' trajectories, the state space and the length of the learning episodes. These bounds under the assumption of consistency and compatibility are proven to be stronger than a "naive" reduction of this cooperation problem to one of Imitation Learning.

My thesis then shows that such collaborations of AI Agents in Alignment with human goals can have real-world applications like with Augmented Reality and Self-Driving Cars. Multimodal

vision-language AI Agents can assist humans proactively by determining when and how the AI Agent will autonomously intervene in real-time to cooperatively solve day-to-day tasks. Augmented Reality (AR) gadgets with distributed edge computing use cases, be it a smartphone or a wearable device, can lead to a major improvement of the user experience in solving procedural day-to-day tasks by introducing egocentric multimodal (audio and video) observational capabilities to AI agents. These AR capabilities help the AI Agents to see and listen to users' actions, thus relating to multimodal capabilities of human users. Current AI Agents, be it Large Language Models (LLMs) or Multimodal Vision-Language Models (VLMs) are mostly reactive in nature, where the AI models cannot take an action without waiting for the human user's vision-language prompts. Proactivity of AI Agents helps the human users to detect and correct any task mistakes by providing more autonomous assistance, encouraging users when they do tasks correctly or simply engaging in conversation with users - akin to a human teaching or helping another human user. I have created a YET to Intervene (YETI) multimodal agent that focuses on the research question of identifying circumstances in real-time that may require the AI agent to proactively intervene. My trained YETI agent can understand when it can intervene in a conversation with human users to help them correct mistakes on tasks, like cooking, using Augmented Reality. YETI learns scene understanding signals based on interpretable notions of Structural Similarity (SSIM) on consecutive observed video frames. It also learns the alignment signal to identify if the video frames corresponding to users' actions on the task are consistent with their expected actions. These signals are used by the AI Agent to determine when it should proactively intervene. I compare the YETI results on the instances of proactive intervention to the HoloAssist multimodal benchmark for an expert agent guiding a user to complete procedural tasks. Control problems for autonomous AI agents, especially safety-critical applications such as autonomous

vehicle control, require robust decision making frameworks to ensure safe navigation in such complex and dynamic environments. This necessitates approaches such as Agentic Model Predictive Control (MPC), which can anticipate future problems and plan for them accordingly. A novel framework has been introduced that integrates MPC with Multimodal VLMs in order to enhance the ability of autonomous vehicles to navigate and respond to real-world scenarios with the ability to take fine-grained actions.

Multi-Agent AI can be pervasive in real world applications, given the foundation of humans and other technological agents to interact with each other, strategize and perform a task. In my thesis, I show that MARL can help to strategize mitigation strategies for climate conservation problems like deforestation mitigation by improving the prediction of deforestation hotspots in Indonesia, one of the two major rainforests in the world. I also share the modeling of another application of Multi-Agent AI collaboration in Supply Chain Orchestration, creating a simulated environment that is cognizant to seasonal demand and cold chains with improved exploration of strategies to maximize profit. I have created a new intrinsic reward signal, helping to save unnecessary interactions among AI Agents planning inventory in Supply Chain warehouses.

These real-world applications motivate the need to understand why the AI Agents behave the way they do which is addressed with Explainable AI (XAI) agents by addressing the question of which XAI methods should be recommended, subject to user agent goals. Explaining the behavior of AI models becomes important in the context of different factors including their training and inference speed that can determine end-users preferring an AI model over another. MARL has been applied to Explainable AI (XAI) problems using a Multi-Agent RecSys to recommend Explainable AI (XAI) outputs for different AI models that can serve the objectives of the model's users for building trustworthy safe AI. Goal-Conditioned RL can be applied to model AI users

learning XAI outcomes as per their preferences. Research on learning to visualize semantic representations satisfying user objectives provides motivation to improve the visualization of XAI methods satisfying the objectives of different users as targets with MARL representations. To represent MARL targets, much of the control problem can be abstracted for deployment in real-world settings like interdiction games, with a much simpler game theoretic problem. Multi-Agent AI algorithms also help to prune AI model parameters across model layers for efficient learning.

As with humans, a large number of AI agents can take a long time to learn strategies jointly. Multi-Agent RL can be pretty slow with increasing scale of agents. To address this, it is shown that the JAXMARL library leverages JAX-enabled hardware acceleration that can make it 12,500x faster over existing libraries in 8 popular MARL environments. The effectiveness of AI Agents can be improved by a combination of the proposed Multi-Agent Reinforcement Learning, Imitation Learning, Model Predictive Control, and Computational Game Theoretic algorithms in solving problems in real world and simulation environments.

MULTI-AGENT AUTONOMOUS DECISION MAKING IN ARTIFICIAL
INTELLIGENCE

by

Saptarashmi Bandyopadhyay

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:

Professor John P. Dickerson, Chair/Advisor
Professor Thomas Goldstein, Co-Advisor
Professor Hal Daume III
Professor David Jacobs
Professor Joseph JaJa
Professor Vint Cerf, External Committee Member

© Copyright by
Saptarashmi Bandyopadhyay
2025

Dedication

I would like to dedicate my Ph.D. degree to my teachers Manjushree Mam, Sayantani Mam, Prof. Ajoy Kumar Ray, my family and my best friend in high school, Soumyadeep. May Soumyadeep Rest in Peace.

Acknowledgments

I would like to express my sincere thanks to my Advisor, John Dickerson, for guiding me throughout these years at University of Maryland College Park. Without the tremendous intellectual freedom that he gave me and the confidence he has in my work, I would not be here today. John has helped me to prosper in my AI research, which has got recognized within the academia, industry and government. He has showed the pathway to dream big and to cross all possible limitations to execute it.

I would like to thank Thomas Goldstein, my Co-Advisor for his continuous support and guidance. My AI research models have been trained with GPU compute at the University of Maryland, Center of Machine Learning where Tom kindly supported my hardware usage. I am of course indebted to Professor Goldstein for his contribution in our joint publications. Whenever I faced any problems, Tom gave me valuable suggestions and guidance. Without his constant motivation and support, it would have been difficult to complete my PhD.

I am very grateful to the guidance and suggestions of Hal Daume III and David Jacobs for being on my PhD committee. Hal's guidance on Fair Machine Learning has inspired my research on real-world applications. I am specially thankful to David Jacobs for his contribution in my multimodal Agentic Model Predictive Control publication. I would like to thank Professor Amol Deshpande, Graduate Chair for his continuous encouragement and support. I would like to acknowledge Professor Aravind Srinivasan, Distinguished University Professor for his support. I

express my gratitude to Ramani Duraiswami for his guidance about pursuing CS research. I have had the unbelievable good fortune of working with brilliant people at the University of Maryland. Their support and expertise helped me to learn more and contribute more for this thesis.

I acknowledge the contribution of Professor Joseph JaJa who has given his valuable advise on my PhD work several times, specially in the area of Explainable AI research.

It was beyond my imagination to have the opportunity to get guidance from the Father of the Internet, Professor Vint Cerf, Google LLC. I must acknowledge as he provided his valuable insight on the different paradigms of my PhD research.

I would like to thank Andrea Colaco, Google for guiding me in the multimodal project YETI - YET to Intervene. In spite of her busy schedule, she always provided me time whenever I needed her guidance. I thank Lin Li, Google for her continuous support and guidance in my AR/VR project. I specially thank Vikas Bahirwani, Lavisha Aggarwal, Bhanu Prakash Reddy Guda for their continuous support in my AR/VR project.

During the summer of 2023, I had the golden opportunity to intern as Google AI Resident at Google X, Mountain View. I learned a lot from my managers Randell Cotta, Lead ML Lead at Google X and Geoff Anders, Data Scientist from Google X group and they are always open to my daringly bold ideas leading to quicker prototypes and focusing on feasible prototypes. Vikranth Dwaracherla and Marc Lanctot from Google DeepMind had from the very beginning kept me aspirational in my research thinking. Jeff Dean, Chief Scientist at Google DeepMind and Google Research; Oriol Vinyals, VP of Research & Deep Learning Lead, Google DeepMind and Astro Teller, CEO of Google X, inspired me in providing the industry-academia flavor in my research work and gave me great feedback after I presented my research to them while at Google X. I must give credit to everyone at Google X, Google DeepMind and Google Research in particular, those

who worked closest with me and those whose open source research inspired my open source PhD research on Multi-Agent AI.

I would like to express my thanks and gratitude to Professor William Higgins, Distinguished Professor, Pennsylvania State University, my MS advisor who inspired me to work in the area of multimodal image analysis. I would like to thank Professor Joseph Genabith, DFKI and Saarland University under whose guidance my international internship during undergraduate Engineering course took off.

I would like to acknowledge Padma Shree Professor Ajoy Kumar Ray, IIT KGP, Former Director IEST who introduced me to the domain of research, focusing on AI and Computer Vision. He ignited my mind to dream big and explore interdisciplinary research solutions. I had my first international research publication with Professor Ray and my friend Soumyadeep.

I am grateful to Robert Loftin from University of Sheffield for helping me juggle theoretical problems and apply theory in empirical research. During the development of inter-institutional JAXMARL project, I had the opportunity to work with Jakob Foerster from University of Oxford who has strongly supported my Multi-Agent AI research and other researchers from Oxford University, Google DeepMind, Meta AI, Waymo UK, and University College London.

I would like to thank Eric Krokos and Sara Savidge of DoD/IC for collaborating with us in the funded project on Recommendation Systems on which I have enjoyed working as the lead PhD student researcher. Together, we have developed and published the Multi-Agent Recommendation System for Explainable AI.

I would like to thank Tom Hurst and Migo Gui at the University of Maryland, Computer Science Graduate Office and the office of the Registrar for helping me whenever required.

My advisor John Dickerson motivated me to create the UMD MARL Reading Group Sem-

inar that has evolved into the AI Agents Reading Group through which I organized regular research talks by reputed academicians like Turing Awardees and industry researchers from Google DeepMind, Microsoft Research, OpenAI, HuggingFace etc. I had the opportunity to collaborate with some of our reputed speakers and also to closely learn and interact with them. Their advice at several instances helped me to learn the state-of-the-art in MARL and integrate these into my PhD research. I would like to thank Turing Awardee Leslie Valiant for giving me grand opportunity to interact with him regarding my topic.

I am glad to thank my peers John (Jack) Cole, Daniel Smolyak, Ryan Sullivan, Jingling Li, Arundhati Banerjee, Anirudh Satheesh, Samuel Dooley, Christine Herlihy, Marina Knittel, Deepthi Raghunandan and others who helped center my work around Multi-Agent AI issues. Over these past years, many bright young School students and undergraduate students with whom I have worked with inspired me and helped me to many fruitful insights. I wish to thank my parents and my friends for their love and support. I would like to acknowledge nature and climate, specially the Himalayas mountains back home, for having motivated my real world research as my experience as hiker and cross-country runner motivates my passion to exploring environments, be it in nature or in AI and solving major challenges.

Table of Contents

Dedication	ii
Acknowledgements	iii
Table of Contents	vii
List of Tables	xii
List of Figures	xiv
List of Abbreviations	xvii
Chapter 1: Introduction	1
1.1 Alignment and Social Cooperation of Autonomous AI Agents	2
1.2 Learning to Cooperate in Populations of Socially Rational Agents	3
1.3 Social Cooperation in Conversational AI Agents	4
1.4 Multimodal Vision-Language Proactive AI Agents for Augmented Reality	4
1.5 Model Predictive Control to Guide Self-Driving Cars	5
1.6 Other Real World Applications of Multi-Agent AI	6
1.7 Multi-Agent Explainable AI	6
1.8 Speedup and Scalability for Efficient Multi-Agent AI	7
Chapter 2: Alignment and Social Cooperation of Autonomous AI Agents	8
2.1 Multi-Agent Alignment	8
2.1.1 Introduction	9
2.1.2 Alignment of LLMs	10
2.1.3 Alignment as a Cooperative Game	17
2.1.4 Cooperative Alignment for LLMs	18
2.1.5 Alignment as Ad Hoc Coordination	19
2.1.6 Formalizing AI Agency	21
2.1.7 Artificial Social Intelligence	22
2.1.8 Multi-Agent Alignment Examples	25
2.1.9 Impact of Artificial Social Intelligence	31
2.1.10 Comparison of Ad Hoc Coordination for AI Alignment with Self-Play	32
2.1.11 Challenges and Open Questions of Multi-Agent AI Alignment	33
2.1.12 Conclusion of Multi-Agent Alignment to Finetune LLMs	34
2.1.13 Limitations and Risks of Multi-Agent Alignment for LLM Finetuning	35

2.2	Social Cooperation of Rational AI Agents	35
2.2.1	Introduction	36
2.2.2	Example of Factory Robot Agents	37
2.2.3	Preliminaries of Social Intelligence	38
2.2.4	Consistent Social Intelligence	40
2.2.5	Cooperative Compatibility	41
2.2.6	Socially Intelligent Agents	43
2.2.7	Social Cooperation	47
2.2.8	A Simplified Model for Social Cooperation	49
2.2.9	Feasible Cooperation	55
2.2.10	Type Inference	56
2.2.11	Conclusion & Major Takeaways of Social Cooperation	57
2.3	Assisted Exploration of Socially Cooperative AI Agents for Alignment	60
2.3.1	Introduction	61
2.3.2	Assistive Markov Decision Processes	61
2.3.3	Noisy Model-Based Learners	63
2.3.4	On-Policy Learners	65
2.3.5	Assisted Exploration	65
Chapter 3:	Learning to Cooperate in Populations of Socially Rational Agents	68
3.1	Introduction	69
3.2	Motivation	72
3.3	Preliminaries	74
3.3.1	Socially Intelligent Agents	74
3.4	Learning to Cooperate	74
3.4.1	Altruistic Regret	75
3.4.2	Consistency without Compatibility	77
3.4.3	Compatibility without Consistency	78
3.5	Lower Bound for Socially Intelligent Populations	80
3.6	Upper Bound for Socially Intelligent Populations	81
3.7	Use Cases of Socially Intelligent Agents	89
3.7.1	AI Health Coach	89
3.7.2	Human-Robot Cooperation in a Factory	90
3.8	Related Works	91
3.9	Conclusion	94
Chapter 4:	Social Cooperation in Conversational AI Agents	96
4.1	Introduction	96
4.2	Preliminaries	98
4.3	Social Intelligence	98
4.3.1	Theoretical Ablations	99
4.3.2	Upper Bound for Social Cooperation	101
4.4	Conclusion	103
Chapter 5:	Multimodal Vision-Language Proactive AI agents for Augmented Reality	105

5.1	Introduction	106
5.2	Related Works	109
5.2.1	Egocentric Interaction Datasets	109
5.2.2	Proactive AI Agents and Communication	110
5.2.3	Language Models for Proactive Assistance	110
5.2.4	Procedural Mistake Detection	111
5.2.5	Action Detection	112
5.3	Methodology	113
5.3.1	Proactive Augmented Reality Interaction Data of Cooperative Agents	113
5.3.2	Generation from Multimodal VLM	115
5.3.3	Alignment with Changing Object Count	116
5.3.4	Spatio-temporal Signal Generation	118
5.3.5	Structural Similarity	118
5.3.6	Proactive Interactions and Interventions	119
5.3.7	YETI Proactive Agent Intervention Algorithm	120
5.4	Experiments	122
5.4.1	Experimental Settings	123
5.4.2	Metrics	124
5.4.3	Results	126
5.5	Detailed Analysis of Results	127
5.6	Comparative Analysis of YETI	129
5.6.1	Comparison with other Classifier Models	129
5.6.2	Comparison with Implicit Expert-User Agent Setting	130
5.7	Ablation Studies	131
5.7.1	Agent Conversation Interval	131
5.7.2	Extrema Range	132
5.7.3	SSIM Thresholding	134
5.7.4	Episode Length	135
5.8	More Use Cases for Proactive Intervention Detection	135
5.9	Conclusion and Future Work	138
Chapter 6: Model Predictive Control to Guide Self-Driving Cars		141
6.1	Introduction	142
6.2	Background	144
6.3	General Approach: VLM-MPCs for Autonomous Vehicles	145
6.3.1	Conceptual Framework	145
6.3.2	Scene Understanding using VLMs	146
6.3.3	Semantic Querying for Object Attributes	147
6.3.4	Integration of Semantic Insights into MPC	148
6.3.5	Real-time Processing and Decision Making	148
6.3.6	Safety-Centric Approach	149
6.3.7	Adaptability and Generalization	149
6.3.8	Human-AI Collaboration	150
6.4	Conclusion and Call to Action	150

Chapter 7:	Other Real World Applications of Multi-Agent AI	152
7.1	Multi-Agent AI for Climate Conservation: Mitigating Deforestation	152
7.1.1	Preventing Deforestation: Modeling and Prediction of Vulnerabilities in Forest Conservation	153
7.1.2	Related Work	156
7.1.3	Dataset	161
7.1.4	Methods	164
7.1.5	Results	168
7.1.6	Conclusion	171
7.2	AI Agents to Autonomously Orchestrate in Supply Chains	172
7.2.1	Introduction	174
7.2.2	Related Work	178
7.2.3	Multi-Agent RL Algorithms for Cooperation in Supply Chains	181
7.2.4	RL Algorithms with Base Stock Constraints	187
7.2.5	Multi-Agent Reinforcement Learning (MARL)	188
7.2.6	Dec-POMDP for Supply Chains	188
7.2.7	Multi-Agent Supply Chain Inventory Management Environment	189
7.2.8	Generalized Multi-Agent Learning with Internal Rewards	195
7.2.9	Results	197
7.2.10	Discussion	202
7.2.11	Conclusion and Future Work	203
7.3	LLM Agents	204
7.3.1	Robustness of LLMs Generating AI Explanations with Zero-Shot and Few-Shot Prompting	205
7.3.2	Personalizable LLM Agents	212
7.3.3	Conclusion	215
Chapter 8:	Multi-Agent Explainable AI	216
8.1	Goal-Conditioned Reinforcement Learning to Recommend AI Explanations	216
8.1.1	Introduction	216
8.1.2	Offline Explanations Dataset	218
8.1.3	Goal-Conditioned Recommendations	219
8.2	Multi-Agent Recommender System for Explainable AI	221
8.2.1	Introduction	222
8.2.2	Use Cases of Multi-Agent XAI RecSys	226
8.2.3	Related Works	228
8.2.4	<i>Review of Multi-Agent Reinforcement Learning</i>	230
8.2.5	XAI Methods and their Properties	235
8.2.6	Multi-Agent XAI Recommender System	237
8.2.7	VDN based Recommendation Results	249
8.2.8	Conclusions and Future Work	250
8.3	Explainable Visualizations	252
8.3.1	Graphical Display of Embeddings	253
8.3.2	Visualizations to Explore Word Embeddings	253
8.3.3	Conclusion & Takeaways to Visualize XAI Outputs	258

8.4	Target Representations in Multi-Agent Reinforcement Learning	258
8.4.1	Introduction	259
8.4.2	Stackelberg Security Games	259
8.4.3	Targets for Competency Self-assessment	260
8.4.4	Modeling SSGs in Reinforcement Learning	261
8.4.5	Target Representation to Assess RL for SSGs	264
8.4.6	Abstracting Targets	268
8.4.7	Conclusion	268
Chapter 9:	Speedup and Scalability for Efficient Multi-Agent AI	269
9.1	JAXMARL: Multi-Agent RL Environments and Algorithms in JAX	269
9.2	Introduction	270
9.3	Background	274
9.3.1	Hardware Accelerated Environments	274
9.4	JAXMARL	274
9.4.1	API	275
9.4.2	Environments	276
9.4.3	Algorithms	280
9.5	Results	281
9.5.1	Environment Speed	282
9.5.2	Algorithm Speed	283
9.5.3	Algorithm Correctness	284
9.5.4	Environment Correctness	285
9.6	Evaluation Recommendations	287
9.7	Related Work	287
9.8	Conclusion	289
9.9	Appendix A: Further Details on Multi-Agent Environments	290
9.9.1	Coin Game	290
9.9.2	Spatial-Temporal Representations of Matrix Games (STORM)	290
9.9.3	Hanabi	291
9.10	Appendix B: Value-Based MARL Methods and Implementation details	292
9.11	Appendix C: Training Results	293
9.11.1	MABrax	293
9.11.2	MPE	294
9.12	Appendix D: Hyperparameters	295
	Bibliography	297

List of Tables

2.1	A simple fully-cooperative 2x2 matrix game.	42
3.1	Payoff matrix for a repeated two-player game with an experienced surgeon (row) and inexperienced surgeon (column). The best cooperative outcome is achieved when experienced surgeon slows down to match the inexperienced.	73
3.2	A class of games where an agent's payoff depends only on its partner.	78
5.1	HoloAssist Feature sizes scaled with YETI Features	108
5.2	Examples of Proactive Interactions and Proactive Interventions in HoloAssist. [17]	120
5.3	Hyperparameters used in YETI experiments.	123
5.4	Performance of YETI compared to HoloAssist in detecting proactive interventions.	124
5.5	Evaluation of YETI on the proactive interaction detection setting which encompasses the proactive intervention benchmark	124
5.6	Training and Test splits of Image Frames with Distributions of Proactive Interactions, Proactive Interventions and 3 Proactive Intervention types	130
5.7	Accuracy of Classification Models in Proactivity Prediction.	130
5.8	Evaluation of YETI compared to HoloAssist and Implicit Interventions	131
5.9	Evaluation of YETI compared to HoloAssist and Implicit Interactions	131
5.10	Evaluation of YETI for Agent Conversation Intervals for proactive interventions .	132
5.11	Comparison over different Agent Conversation interval sizes for proactive interactions.	133
5.12	Evaluation of YETI for Extrema Ranges for proactive interventions	133
5.13	Comparison over different Extrema ranges.	133
5.14	Evaluation of YETI for SSIM Thresholding for proactive interventions	134
5.15	Comparison over different SSIM Thresholding levels for proactive interactions. .	134
7.1	Performance of different versions of the deforestation vulnerability detection model on the test data.	162
7.2	Comparison of Algorithms Covered by STEIMAC (Ours) and MABIM	176
7.3	Feature Comparison between STEIMAC and MABIM.	198
7.4	Training times for each algorithm under different numbers of echelons for 200 SKUs in each warehouse.	201
7.5	Statistics of Two Training & Testing Datasets for Random Forest Classification .	207
7.6	Accuracy of LLM labelling with different prompting strategies	211
8.1	Statistics of Training & Testing Datasets for two AI model goals	218
8.2	Summary of Goal-conditioned Reinforcement Learning RecSys for Explanations	219
8.3	Statistics of Training & Testing Datasets for two AI Models	227

9.1	Benchmark results for JAX-based MARL environments (steps-per-second) when taking random actions. All environments are significantly faster than existing CPU implementations. [256]	278
9.2	Recommended Minimal Environment Evaluations for different research settings	286
9.3	MABrax Hyperparameters, where – indicates repeated parameters [256]	295
9.4	Hyperparameters for MPE IPPO	296
9.5	Hyperparameters for Hanabi IPPO [256]	296
9.6	Hyperparameters for Overcooked IPPO	296
9.7	Hyperparameters for MPE Q -Learning Algorithms [256]	296

List of Figures

2.1	Human-LLM conversation on 3 digit multiplication starting on the left and continuing on the right	28
5.1	Overview of the YETI framework for detecting proactive interactions or interventions in Augmented Reality. YETI uses lightweight signals generated in real-time, allowing timely decision making for providing assistance to the user [17].	107
5.2	Distribution of Alignment Signal	117
5.3	A plot of the alignment signal over time for a video, accompanied by two consecutive video frames where a change in object count occurs. YETI detects a proactive intervention to help the user replace a mechanical belt on Frame 37 after observing the alignment signal from Frame 36. [17]	117
5.4	SSIM signal is filtering proactive interventions by YETI Agent in an image frame (time instance) of a video observing a procedural task on how to assemble a RAM computer. Proactive intervention happens at the 98th second in Frame 98	118
5.5	Proactive Intervention Detection for Coffee Making Task. [17]	126
5.6	Intervention Detection for Changing Electric Circuit Task.	137
6.1	Overview of the decision making process for the VLM-MPC system for Autonomous Vehicle Control.	146
6.2	PaliGemma 3B can accurately identify and detect moving cars in a video.	147
7.1	Tree cover loss in Indonesia from 2000 to 2020.	153
7.2	Confusion matrix for the base model with terrain features and BoostIT	169
7.3	Visualization of the decision tree learned for the base model with terrain features and BoostIT (Please zoom in to see the individual nodes of the decision tree more clearly)	170
7.4	Illustration of the bus network setting. Each agent represents an SKU in a warehouse, with interactions occurring between corresponding SKUs across warehouses. Agents’ actions involve requesting specific replenishment quantities from upstream warehouses, while warehouses sell a portion of each SKU’s inventory to downstream warehouses based on the replenishment order.	174
7.5	Test reward with and without seasonality data using the (a) IPPO algorithm and (b) IPPO base stock algorithm.	192
7.6	Test reward with and without seasonality data using the (a) QTRAN algorithm and (b) QTRAN base stock algorithm.	193

7.7	Performance comparison of reinforcement learning algorithms across different settings. (a) Standard Setting : Standard MARL settings as in MABIM [335]. (b) Seasonality Setting : Performance under settings with seasonality patterns. (c) Outlier Setting : Performance under settings with outlier data. (d) Standard Setting with Cold Chain SKUs : Standard setting with the inclusion of cold chain SKUs. (e) Seasonality Setting with Cold Chain SKUs : Seasonality setting with the inclusion of cold chain SKUs. (f) Outlier Setting with Cold Chain SKUs : Outlier setting with the inclusion of cold chain SKUs. Each subfigure shows the average performance and variability of the algorithms under the corresponding scenario.	197
7.8	Train performance comparison of MARL algorithms with different intrinsic reward algorithms. (a) Standard Setting : Train Performance of Standard MARL setting as in MABIM [335]. (b) ELIGN Setting : Train Performance of MARL algorithms with ELIGN intrinsic rewards. (c) CURLS Setting : Train Performance of MARL algorithms with CURLS intrinsic rewards.	199
7.9	Test performance comparison of MARL algorithms with different intrinsic reward algorithms. (a) Standard Setting : Test Performance of Standard MARL setting as in MABIM [335]. (b) ELIGN Setting : Test Performance of MARL algorithms with ELIGN intrinsic rewards. (c) CURLS Setting : Test Performance of MARL algorithms with CURLS intrinsic rewards.	200
7.10	(a) LIME explanation and (b) SHAP explanation for a test sample in the Couples dataset	208
7.11	An example of a zero-shot prompt for Flan-T5-XL	209
7.12	An example of a few-shot prompt for Flan-T5-XL	210
7.13	Confusion Matrices for the Couples dataset comparing zero shot and few shot prompting	210
7.14	The rows of a table of the most important tokens in the prompt impacting the generation as scored by the Input X Gradient method [277]	212
8.1	(a) LIME explanation and (b) SHAP explanation for a test sample in the couples dataset	218
8.2	Average Episode Reward across time-steps for two goals while simulating with SlateQ RecSys in comparison with a Q-learning approach	220
8.3	Multi-Agent XAI Recommender System Architecture	221
8.4	Implicit Negotiation between User Agents for XAI Outputs of AI Models	241
8.5	Evaluation Metrics for QMIX Agents and Random Slates	248
8.6	Avg Alignment for QMIX and VDN in comparison to Random strategies	248
8.7	Recommendation Diversity with QMIX Agents	249
8.8	Evaluation Metrics for VDN Agents and Random Baseline	250
8.9	Recommendation Diversity with VDN Agents	250
8.10	Embedding vectors for three male and three female words. Component 76, marked by a vertical black line, is positive for male words and negative for female words.	254

8.11	Embedding vectors for three words associated with adulthood and three words associated with youth. Component 14, marked by a vertical black line, is positive for the adult words and negative for the youth words.	254
8.12	Words plotted in 3D semantic space. Male words appear in the positive (left) half of the x-axis; female words in the negative (right) half. Adult words are in the positive (top) half of the y-axis; youth words in the negative (bottom) half.	255
8.13	A rotated view of the scatter plot showing distance along the third, residual dimension, which is labeled Genderless/Ageless. The contrast pair words, which define the residual dimension, have residuals close to zero. “Refrigerator”, which is semantically unrelated to the contrast words, is located a substantial distance from them (high residual).	255
8.14	Hovering over a point in the scatter plot produces a display of the word and its 10 nearest neighbors.	256
8.15	Layout of the developed interactive embedding visualization demo. On the left-side is a zoomable and rotatable 3D scatter plot. On the right is a display of selected embedding vectors. The text box at the bottom allows entry of new words.	256
9.1	JAXMARL was built with the purpose of being an easy-to-use MARL library that does not compromise on efficiency and variety.	271
9.2	JAXMARL environments. Vectorized implementations of a wide range of environments are provided from different MARL settings.	272
9.3	An example use of JAXMARL’s adaptable and intuitive API.	275
9.4	JAXMARL provides a considerable speedup compared to CPU-based environments as shown in these four examples [256].	281
9.5	Speed and Performance (Return, abbreviated as Ret.) in JAXMARL compared to MARLLIB and PyMARL in MPE. Return results were averaged across 3 seeds. Performance results show 1 seed collected on the hardware described in Section 9.5.1.	282
9.6	Performance and speed of JAXMARL <i>Q</i> -Learning baselines compared to PyMARL on MPE. These implementations match PyMARL’s returns, while being over 2000x faster to train	284
9.7	JAXMARL IPPO baseline results. These results correspond to similar baselines and therefore demonstrate the correctness of JAXMARL implementations.	286
9.8	Performance of IPPO on MABrax Tasks [256]	294
9.9	<i>Q</i> -Learning Baselines in all MPE scenarios. Where no algorithm names are given, the results represent IQL. [256]	294

List of Abbreviations

AC	Approximately Compatible
AGI	Artificial General Intelligence
AI	Artificial Intelligence
AHT	Ad Hoc Teamwork
API	Application Programming Interface
AR	Augmented Reality
CIRL	Cooperative Inverse Reinforcement Learning
CPU	Central Processing Unit
CTDE	Centralized Training with Decentralized Execution
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q Learning
DEC-POMDP	Decentralized Partially Observable Markov Decision Process
DQN	Deep Q-Network
ELIGN	Expectation Alignment
GAN	Generative Adversarial Network
GCRL	Goal-Conditioned Reinforcement Learning
GIS	Geographic Information System
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HLE	Hanabi Learning Environment
IPPO	Independent Proximal Policy Optimization
IQL	Independent Q-Learning
IRL	Inverse Reinforcement Learning
ISE	Inducible Stackelberg Equilibrium
JIT	Just-In-Time
LIME	Locally Interpretable Model-Agnostic Explanations
LLM	Large Language Model
MAB	Multi-Armed Bandit
MABIM	Multi-Agent Benchmark for Inventory Management
MAPPO	Multi-Agent Proximal Policy Optimization
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent Systems
MDP	Markov Decision Processes
MPC	Model Predictive Control

MPE	Multi-Agent Particle Environments
MPSE	Markov Perfect Stackelberg Equilibrium
NE	Nash Equilibria
PEO	Polynomially Empirically Optimal
POMDP	Partially Observable Markov Decision Process
PONE	Pareto-Optimal Nash Equilibrium
PoS	Part-of-Speech
POSG	Partially Observable Stochastic Game
QMIX	QMIX
RL	Reinforcement Learning
RLAIF	Reinforcement Learning from Artificial Intelligence Feedback
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network
RT	Robotic Transformer
SHAP	Shapley Additive Explanations
SI	Socially Intelligent
SKU	Stock-Keeping Unit
SMAC	StarCraft Multi-Agent Challenge
SSG	Stackelberg Security Game
SSIM	Structural Similarity
SSE	Strong Stackelberg Equilibrium
STEIMAC	Seasonal Trends for Efficient Inventory Management with Multi-Agent Coalitions
STORM	Spatial-Temporal Representations of Matrix Games
TPU	Tensor Processing Unit
TRPO	Trust Region Policy Optimization
VDN	Value Decomposition Networks
VLM	Vision-Language Model
WSE	Weak Stackelberg Equilibrium
XAI	Explainable AI
XLA	Accelerated Linear Algebra
YETI	YET to Intervene
ZSC	Zero-Shot Coordination

Chapter 1: Introduction

Multi-Agent Autonomous Decision Making, especially Multi-Agent Reinforcement Learning (MARL), is an emerging area of Artificial Intelligence (AI) where autonomous agents interact with each other, fostering competition and/or cooperation in real world problems like climate change, supply chains, self-driving cars, sports, interdiction games, war games among other uses.

My PhD research has been arranged on the research works that I have already carried out and published along with formal research in the field of AI Agents as discussed below. My thesis research addresses challenges in fundamental Computer Science domains with the opportunity of AI Agents, including Robotic Agents, Augmented Reality with AI Agents, Self-Driving Cars, Climate Conservation, Supply Chain Orchestration and Explainable AI Agents. My PhD thesis research offers solutions to these real-world as well as simulation problems with an Algorithmic paradigm of Agentic AI unifying relevant Multi-Agent Reinforcement Learning, Imitation Learning, Self-Supervised Learning, Game Theoretic and Model Predictive Control Algorithms.

1.1 Alignment and Social Cooperation of Autonomous AI Agents

Human-AI Value Alignment has emerged as a central challenge in the rapid deployment of Artificial Intelligence (AI). In many applications like Large Language Models (LLMs) or Vision Language Models (VLMs), the goals and constraints for the AI model's desired behavior are known only to a (potentially very small) group of humans, during large-scale pre-training of these models. The objective then is to develop mechanisms that allow humans to communicate these goals both efficiently and reliably to AI agents like LLMs prior to their deployment. This requires explicitly reasoning about human strategies for shaping the behavior of AI agents, and how these strategies are derived from the humans' prior beliefs about the AI's own learning process. In my PhD thesis, the alignment problem has been viewed in Chapter 2 from the perspective of Multi-Agent Systems (MAS) by formalizing AI Agency and introducing the Artificial Social Intelligence Framework (ASoCI) in Section 2.1. Existing research shows that modeling just good rewards are not sufficient to model this zero-shot coordination problem as it can solve capability mis-generalization without addressing goal mis-generalization. The modeling of alignment as ad-hoc coordination in the ASoCI framework helps to address this formally while ensuring an efficient alignment framework for Multi-Agent cooperation between an AI Agent and any agent in a socially intelligence population, which existing works do not address. I introduce the game-theoretic background of Social Intelligence with Consistency and Compatibility in Section 2.2 which helps to set-up a notion of feasible cooperation when the user goals (types) are initially unknown or private. To make sure that an AI Agent can effectively assist users with efficient

exploration, a theoretical framework for Assisted Exploration of Socially Cooperative Agents to ensure alignment has been provided in Section 2.3.

1.2 Learning to Cooperate in Populations of Socially Rational Agents

It is critical for an AI which is deployed in the real to cooperate with other humans and AI agents if that AI is to successfully align to many applications, especially when the goals of the other agents are initially unknown as shown in Chapter 2. A natural aim is to design AI agents that can reliably adapt to their current socially intelligent human partner's strategy. The human partner is likely to be just as adaptable as the AI itself and this may render cooperation impossible. An environment has been considered in Chapter 3 where an AI must learn to cooperate with an existing population of socially intelligent agents that are individually rational and at the same time reliably cooperating with any member of the population in a finitely repeated two-player general-sum Bayesian game. The challenge for the AI is to generalize from previous interactions that allow it to cooperate with a new partner drawn from this population. My thesis provides lower bounds on sample complexity for such cooperative settings, highlighting the need for sufficient data covering initial interaction histories (like handshake protocols). My thesis also provides upper bounds for learning near-optimal cooperation meta-strategies in socially intelligent populations, which are introduced as imitate-then-commit strategies. The objective of the problem is modeled as minimizing Altruistic Regret (partner's regret relative to their Pareto-Optimal Nash Equilibrium (PONE)). These bounds can be much stronger than the sample complexity bounds

for "naive" methods that treat the zero-shot coordination problem of the AI Agent with a member of the socially intelligent population as one of imitation learning or reinforcement learning.

1.3 Social Cooperation in Conversational AI Agents

The theoretical guarantees of Social Cooperation in Chapter 3 motivates investigation of using these bounds to help train conversational AI Agents in Chapter 4. These AI Agents have to maintain long-term interactions and cooperation with collaborating human agents whose goals are unknown to begin with. Reducing this cooperation problem to be done with imitation learning is infeasible as imitation learning has statistical challenges when state spaces get too large and time horizons get too long. These formal guarantees can be used for conversational AI Agents. cooperating with socially intelligent populations with the assumptions that 1) members of the population are individually rational, and 2) any two agents in the cooperative setting can achieve performance comparable to a Pareto-optimal Nash equilibrium. These guarantees are in the offline cooperation setting where the agents must cooperate with unseen settings in the population to strategize against and manipulate no-regret policies. The complexity analysis and formal bounds can be helpful to ensure alignment of such conversational AI Agents.

1.4 Multimodal Vision-Language Proactive AI Agents for Augmented Reality

With the above introduction of Artificial Social Intelligence and Social Cooperation theoretically, it is imperative to explore such interactions in the real world which deals with Mul-

timodal Vision-Language AI Agents that assist humans in their daily tasks. As Multimodal AI Agents have grown in popularity, one limitation they face is being reactive to user needs by waiting for users to ask questions, rather than proactively anticipating those needs which is often needed for educational or industrial settings. The weak autonomy of reactive AI Agents, needing a human to ask a query, is especially pronounced in smart devices like Augmented Reality environments where the user may be in a wide variety of real-world scenarios where help from the AI Assistant would be helpful but the user Agent either cannot explicitly ask for assistance or is not aware they need assistance. In Chapter 5, I highlight YET to Intervene (YETI) Agents, which detect when and how to proactively intervene to assist users. Compared to the HoloAssist [319] baseline method, YETI has a better performance in online detection of more ground truth proactive interventions with comparable precision and higher recall, setting the stage for multimodal Augmented Reality AI Agents to proactively intervene.

1.5 Model Predictive Control to Guide Self-Driving Cars

Another application is the classical Multi-Agent problem of Autonomous Vehicles, as each vehicle must be aware of pedestrian, obstructions and vehicle agents to operate safely. Model Predictive Control from control theory is a widely used solution to have safety constraints, but fail to generalize in some cases, while modern multimodal AI technology has powerful semantic understanding capabilities but is prone to hallucinations. In Chapter 6 Agentic Multimodal Model Predictive Control, a framework to integrate the two, informing classic Model Predictive Control settings with insights from low-latency Vision-Language Models can be helpful to enable fine-grained control for such agents.

1.6 Other Real World Applications of Multi-Agent AI

More real world use cases of interacting AI Agents are shared in Chapter 7. The present work is focused on the climate conservation problem of deforestation with the goal of learning a MARL environment that can lead to exploring effective strategies to mitigate deforestation. The MARL environment is modeled on the prediction of attacks on tree cover, a green security asset, in sub-national regions of Indonesia using a boosted Decision Tree Classifier, the BoostIT algorithm in Section 7.1. A boosted Decision Tree Classifier takes minimal resources to build, is accurate in its predictions, and is scalable for the sake of expanding on the assumptions made regarding the drivers of deforestation. Such an algorithm can empower communities to manage forest resources effectively. Further real world applications of Multi-Agent AI are being pursued on AI Agents to Autonomously Orchestrate in Supply Chains with the Seasonal Trends for Efficient Inventory management with Multi-Agent Cooperation (STEIMAC) framework in Section 7.2 that considers complexities in the multi-echelon global supply chains like seasonal economic trends, cold-chain shipments, outlier economic trends among others. LLM Agents in Section 7.3 have been trained for downstream tasks like explanation generation while exploring personalization to user's requirements.

1.7 Multi-Agent Explainable AI

For real-world use cases, the ability for AI Agents to make decisions that are explainable and interpretable to User Agents is absolutely essential. At the same time, different User Agents may have different goals and requirements, even when they are collaborating with other User

Agents. This motivates my research on Goal-conditioned RL to recommend AI explanations, a Multi-Agent Recommender System to recommend AI explanations, and Interpretable Visualizations for Explainable AI outputs, and Stackelberg Security Games for interpretable target representations, presented in Chapter 8.

1.8 Speedup and Scalability for Efficient Multi-Agent AI

Interactions between multiple Socially Intelligent Agents, in Social Cooperation or other applications as introduced before, can scale exponentially with increasing number of agents and complex action spaces. A significant bottleneck exists during MARL training where environment execution is CPU-bound and algorithm training is GPU-bound. JAXMARL is the first open-source Multi-Agent AI code base as proposed in Chapter 9 that combines ease-of-use with GPU-enabled efficiency, and supports commonly used MARL environments and popular algorithms. I initiated the inter-disciplinary JAXMARL library project with my coding contributions in JAX-based environments by vectorizing the environment roll-outs and co-locating the agent updates on the same hardware accelerator (the GPU) for some MARL algorithms. Introduction of Agentic Populations in Sections 2 and 3 are relevant to my research here to scale MARL algorithms effectively across agent groups while vectorizing the agent's learned policies. Per-run, JAX-based Multi-Agent AI training pipeline in JAXMARL experiments is up to $12,500\times$ faster than existing approaches, enabling efficient and thorough evaluations. The code base is available in the open-source on Github. Discussions on the work on Speedup and Scalability for Efficient Multi-Agent AI have been discussed in Chapter 9 and [256].

Chapter 2: Alignment and Social Cooperation of Autonomous AI Agents

2.1 Multi-Agent Alignment

Human-AI Value Alignment has emerged as a central challenge in the rapid deployment of Artificial Intelligence (AI). *Mis-Alignment* is an important problem in the development of Large Language Model (LLM) agents in the real world that can reliably cooperate with rational agents like humans. In most applications, the goals and constraints for the LLM's desired behavior are known only to a (potentially very small) group of humans and application-specific LLM Agents are reliant on short-term interactions among humans' as training data to communicate. The objective then is to develop mechanisms that allow humans to communicate these goals both *efficiently* and *reliably* to AI models prior to their deployment. This requires explicitly reasoning about human strategies for shaping the behavior of AI agents, and how these strategies are derived from the humans' prior beliefs about the AI's own learning process. I propose that it is natural to view the alignment problem from the perspective of Multi-Agent Systems (MAS). I advocate for the *Artificial Social Intelligence (ASocI)* framework, which models humans as populations of rational agents and can serve as a framework for Aligning LLM agents with human goals, building on top of fundamental formalization and theoretical bounds provided in Chapter 3. I present a survey of open alignment challenges in the context of finetuning of LLM agents that

cooperate with humans, and in zero-shot learning with LLMs. I then connect these open questions to concepts developed for the ASocI framework, explicitly modeling human agent’s behavior to establish long-term cooperation with other agents (particularly for *ad hoc coordination*). The objective of this Multi-Agent alignment framework is to effectively communicate LLM agents’ goals to humans and vice-versa, which can help in improving alignment of LLM agents.

2.1.1 Introduction

Until recently, most work on machine learning (and artificial intelligence in general) has sidestepped the *alignment* problem, and assumed that the goals of an AI system are well defined for applications like Recommender Systems [16]. With the advent of large language models (LLMs) that have been finetuned with human feedback [229, 232], the process by which human designers and/or users communicate their goals for AI has become an issue of immediate practical importance. Typically, the interpretation of human-generated data has been based on simple, fixed models of the processes by which such data is generated. These models represent assumptions about human behavior that, if mistaken, may lead to the training of AI models like LLMs that are *mis-aligned* with human goals. The use of fixed models therefore encourages practitioners to limit themselves to unambiguous forms of communication, such as binary feedback, that admit safe(r) generative assumptions but convey very little information with each example.

I hypothesize that recent methods and theory for Multi-Agent Systems can be brought to bear on the problem of human-AI alignment, and that alignment issues in the finetuning of LLMs and other large models represent an important future direction for research on topics such as Multi-Agent Reinforcement Learning and Game Theory. It has previously been argued that

alignment problems in general can be thought of as cooperative games [129, 148], but this formulation itself does not address the fundamental question of what *strategy* a human will follow when teaching an AI. Recently, however, there has been substantial progress on the problem of *ad hoc coordination* [213], particularly in the context of Multi-Agent Reinforcement Learning [53, 284, 304]. These methods address (in many different ways) the problem of cooperating with an agent whose strategy is unknown *a priori*. By applying these methods to the alignment problem, rather than committing to a fixed model of human feedback, it may be possible to design (or train) learning strategies that are robust to the different approaches that humans may take when teaching an AI model.

The challenge of finetuning large language models serves as both motivation for this work, and as an illustrative example of the broader alignment problem. I first review approaches to finetuning LLMs, and some of the alignment issues that arise with these methods. I then describe Multi-Agent formalizations of the finetuning problem, and give examples of how specific alignment failures may potentially be addressed using approaches to ad hoc coordination. I conclude with a discussion of the potential challenges of applying existing methods, and the open research questions surrounding Multi-Agent approaches to alignment.

2.1.2 Alignment of LLMs

Alignment of AI Models like Large Language Models (LLMs), Vision Language Models (VLMs), Robotic Transformer (RT) AI Agents among others is a very important objective with large scale deployment of AI to ensure that the AI and the human goals are closely lined up. This opens up the learning problem on how humans can communicate their goals to AI effectively

to help the AI adapt to corrective feedback. An approach to solve alignment of AI models can be with an Empowerment objective. Empowerment can be a local and universal utility function without relying on an infinitely long past experience. This objective function does not require global knowledge and gives localized feedback to motivate agents survive and evolve quickly in a smoother fitness landscape as defined in [162]. The behavioral empowerment hypothesis proposed in [259] discusses about the adaptation among agents brought about by evolution arising from the necessity to maximize empowerment in the absence of specific goals. However such assumptions without the representation of human goals can lead to *mis-aligned* LLMs. [50] discusses the challenges of empowerment objectives to align AI systems. An instrumental convergence to empowerment can lead to challenges of differentiating altruistic and selfish empowerment along with relative computational costs, approximation issues, deceptive alignment, corrigibility, goodharting, identity preservation, coordination advantages and most importantly to empower whom or what among other challenges.

Different general frameworks of Alignment that address challenges originating from the empowerment objective discussed above include:

1. Reinforcement Learning (RL) from Human and AI Feedback in Section 2.1.2
2. Imitation Learning in Section 2.1.2
3. Multi-Agent Reward Guided Alignment in Section 2.1.2
4. Representational Alignment in Section 2.1.2

RL from Human and AI Feedback

The most common approach to finetuning LLMs is RL from Human Feedback (RLHF) which can be expensive to collect human generated data. RLHF can also introduce biases and noise to the LLMs objectives among other challenges [55]. LLMs and other AI models can be finetuned with Reinforcement Learning from Artificial Intelligence Feedback (RLAIF) [13] which can be helpful to make alignment self-supervised and can help to mitigate challenges of scaling Reinforcement Learning from Human Feedback (RLHF) to finetune Large Language Models (LLM) [174]. Scaling supervision may be helpful to oversee the behavior of LLM agents if the supervisor agents' capabilities scale better or similar to the actor agents' capabilities and the supervisor agents are aligned to a problem's goals. A Chain of Hindsight approach to AI Alignment [184] transforms different feedback modalities into a sequence of sentences to finetune LLMs capitalizing on the language comprehension skill of LLMs. This approach can be easy to optimize and can help generalize for specific tasks like summarization and dialog using different types of feedbacks with imperfect reward functions.

Relying extensively on feedback to finetune AI models like LLMs can still be challenging to solve for many problems with openended objectives having ambiguous goal representations where exploration based on internal and external knowledge can be helpful. Reliance on feedback for alignment can cause challenges in the form of misaligned feedback providers aggravated by over-optimization on human feedback, task misgeneralization, distributional challenges, oversight issues, lack of diverse feedback among other issues [55]. RLHF Alignment also does not help in securing against jail-breaking using adversarial prompts including those adversaries which don't need much compute [209]. The simpler adversarial prompts are transferrable which

means that these prompts can be generated by 1 LLM to jail-break another LLM. There are alternative alignment approaches for alignment without feedback like LIMA [358] which finetunes a 65B LLaMA LLM [301] using a supervised loss on just 1000 curated prompts. Curating these prompts, however, involves a great deal of human effort, raising similar issues as preference-based feedback.

Recent work has sought to improve AI safety by modeling Language based games as repeated games with debating AI agents using oracle-based learning paradigms [65] which improves best response of AI models given noisy feedback. However this approach does not investigate general sum games or LLM finetuning. It is also data-dependent on the debate interactions of agents and focuses on a sub-case of LLM agent interactions like debates where a judge agent can adjudicate and stop the debate that may not be useful in generic cases like mathematical reasoning.

Imitation Learning

Value Alignment of AI agents has been modeled as a Cooperative Inverse Reinforcement Learning (CIRL) [129] which is a co-operative partial information game having a human agent and an AI agent, with the AI agent having no knowledge about the human's reward and both agents getting rewards from the human's rewards. Optimal Joint CIRL policies can be calculated using a Partially Observable Markov Decision Process (POMDP) to generate agent behavior like active teaching or active learning which helps to ensure Alignment. Few-shot Imitation Learning can be helpful for AI agents to learn domain general social skills like cultural transmission [34] that can in-turn help agents to acquire, aggregate, update and adapt skills and knowledge along generation. An autonomous approach of real-time imitation of humans without collecting human

feedback can be helpful as one of the stepping stones to align Artificial General Intelligence (AGI).

The Bayesian Inverse Reinforcement Learning (IRL) framework can be generalized to inverse contextual bandits where the expert policy may change over time as the expert learns about the task domain [142]. It appears to marginalize over the the expert's own observations and corresponding beliefs about the task environment, which may change over time. Such Bayesian IRL experiments have been performed for autonomous decision making on organ allocation (liver transplants) with the goal being to understand (and explain) how the decision making process for organ allocation has changed over the past three decades. [117] pursues primarily experimental research on Bayesian IRL in settings where the expert is learning in the environment. An empirical demonstration is provided giving insights that reward inference can actually be more efficient when the expert is learning than when they are simply noisily rational against a known reward function and environment.

Imitation based Alignment can be helpful to simplify finetuning of LLMs instead of finetuning with feedback since imitation does not learn interactions, rather it learns demonstrations. So, there is not much worry about the exponential complexities arising from Multi-Agent interactions. However, there are a few challenges in getting good demonstrations or trajectories of agent behavior.

Multi-Agent Reward Guided Alignment

An important aspect of communicating reward models during the LLM finetuning would be to look into sparse rewards for real world applications. To overcome challenges of training Multi-Agent cooperative AI in a decentralized manner with sparse rewards, a self supervised intrinsic

reward function *ELIGN* Expectation Alignment [199] can be used to train agents for matching their neighbor's expectations better than curiosity-driven exploration. This approach of training Multi-Agent systems is motivated by the idea of self-organization in Zoology and allows agents to learn without any centralized training or external rewards as demonstrated in the Multi-Agent Particle Environments [191] and Google Research Football environments [167]. Learning of such internal rewards can be helpful to guide agents' behavior across different modalities [159], like computing similarity of visual outcomes with natural language instructions adaptively in each time step as the reward signal to address goal mis-generalization. The return-conditioned policies lead to better generalization than text-conditioned policies which can be improved further with finetuning, highlighting the importance of internal reward representations for multiple agents. An alternative idea to guide AI policies outside rewards or demonstrations involves iterative steps of corrections using meta-learning [76] which can help guide AI agents to learn specific skills. Real world applications of LLMs and other AI models can be context dependent where guiding the AI agents' behavior can depend on a hybrid combination of internal reward representations along multiple dimensions or instruction sets with meta-learning depending on the specifications and constraints of the problem being solved by AI.

Representational Alignment

Representations of human goals are essential to communicate reward models for different objectives while finetuning LLMs. Representational alignment involves investigation of agreement among multiple agents and how the agent representations can be updated to better match with other agents, as motivated from diverse research in Machine Learning, Cognitive Science and Neuroscience [287]. Challenges of representational alignment involve the lack of a guarantee

of aligned behavior of agents to solve tasks despite having an agreement of agents, appropriate similarity measures among agent representations, downstream biases among other issues. Alignment can be defined probabilistically on human representations and representational alignment can quantify agreement in alignment among agents including humans which can help an aligned agent to supervise a mis-aligned agent, as shown using information theory [286]. The relationship between human representations and performance of aligned AI agents on few-shot learning tasks can be U-shaped [286] and be robust to adversarial attacks and domain shifts as demonstrated on 491 computer vision tasks. Representational alignment provides an useful framework for my formulation of cooperative alignment games to measure an agreement of the LLM's interpreted feedback representations with the human's goal representations.

Even with different paradigms of AI Alignment, there are open research problems in AI Alignment originating from different failure modes of Alignment which is a research focus to address with Ad Hoc Coordination based Multi-Agent Alignment.

1. Incorrect generalization to new situations
2. Ignoring explicit communication in the data itself
3. Fictitious explanations which can break the ability to verify alignment of AI agents
4. Self-empowerment goals
5. Hallucination which can originate from "embodiment" representations

2.1.3 Alignment as a Cooperative Game

Implicit in most approaches to learning from human feedback (or demonstrations, instructions, etc.) is an assumption about the generative process from which such data arises, and how that process depends on the goals the AI is expected to learn to achieve.¹ Many of the alignment challenges discussed above may be addressed by treating this generative process as a *strategy* chosen by the human teacher(s) to shape the AI’s behavior. This approach can be identified by two key features: 1) explicitly reasoning about the space of *plausible* human strategies, and 2) assuming that human strategies are (at least approximately) *rational* with respect to their goals for the AI’s behavior. Previous works [129, 134, 148, 188] have shown the value of explicitly reasoning about human strategies in interactive learning. [134] in particular finds that human demonstrations are often more consistent with an intentional teaching strategy than with the target behavior itself. In [134] it was shown that many different models of human feedback and demonstration can be reformulated as softmax-rational strategies given the underlying reward function for the task.

To make things more concrete, I will use the *Cooperative Inverse Reinforcement Learning* (CIRL) formalism [129], which models interactive learning as a two-player, fully cooperative game with imperfect information. An instance of CIRL is defined by a tuple $M = \{S, A^H, A^{AI}, T, \Theta, R, P_0, H\}$, where S is the joint state space, A^H and A^{AI} are the action spaces available to the human and AI respectively and $T : S \times A^H \times A^{AI} \mapsto \Delta(S)$ is the transition kernel. The key feature of the CIRL model is the space Θ of possible reward function parameters (or “types” in the language of Bayesian games). The joint reward function

¹In RLHF, this would be the reward-dependent likelihood over pairwise preferences.

$R : S \times A^H \times A^{AI} \times \Theta \mapsto \mathfrak{R}$ is parameterized by the current type $\theta \in \Theta$, which is only ever directly observed by the human. At the start of the game, the initial state and type θ are sampled from the prior P_0 , and the players are allowed to interact for H time steps. The game M (including the prior P_0) is assumed to be commonly known to both the human and the AI. Here I refer to a specific instance M of the CIRL model as a *cooperative alignment game*, and let π^H and π^{AI} correspond to the human and AI respective strategies. As each player may be uncertain of its partner’s strategy (and the AI is uncertain of the type θ), I assume that each player’s strategy depends on the entire *history* of states and actions.

2.1.4 Cooperative Alignment for LLMs

In my motivating context of finetuning large language models, the AI’s strategy π^{AI} captures the entire learning pipeline, including the generation of candidate responses, and the training of both the reward model (if one is present) and the language model itself. The problem of designing an algorithm for learning from a given form of human interaction then corresponds to that of finding a “good” strategy for the AI in the corresponding cooperative alignment game. As an example, a cooperative alignment game can be defined for an online version of RLHF with a single teacher. Here the type space θ would be the parameter space of the reward model, while the state space S would consist of possible prompt strings, sampled i.i.d. from some fixed distribution. The AI’s action space A^{AI} would consist of the space of k -tuples over response strings, while the human’s action space A^H would consist of possible preference orderings over the latest set of responses. If the reward function is computed based on the quality of the AI’s first response to each prompt, then the shared goal for both the human and AI is to maximize the quality of the

AI’s responses over a series of H prompts.

The standard RLHF [70] paradigm makes the implicit assumption that the human’s strategy π^H ranks responses based purely on their quality under the reward model $R(\cdot; \theta)$. Given the complexity of the response space, however, humans might find it more efficient to rank responses based on how well they address some known deficiency of the model (e.g., consistency of past vs. present tense) rather than their overall quality. If the AI’s strategy fails to take this into account, such examples may cause it to “unlearn” if they contradict previous examples that were ranked based on reward alone. On the other hand, if the AI knows the human is using such a strategy, it can not only resolve these apparent contradictions, but actively cooperate with the human teacher by providing responses that vary along a single dimension about which the AI is uncertain. The idea that "preferences" expressed by a human may depend on their teaching strategy can appear counterintuitive. Nonetheless, empirical studies have demonstrated this phenomenon when human subjects are asked to teach AI using feedback [188, 200] or demonstration [134]. [134] shows that humans often provide demonstrations that are sub-optimal, but more informative to the AI.

2.1.5 Alignment as Ad Hoc Coordination

The example above highlights the central challenge with the *strategic* approach to alignment, that is, the uncertainty as to what teaching strategy the human will employ when teaching the AI. In recent years, substantial progress has been made in addressing this same problem in the context of human-AI cooperation. Methods for *ad hoc coordination* (also referred to as *ad hoc teamwork* and *zero-shot coordination*) seek strategies that can cooperate with partners whose

strategies are unknown *a priori* [213].

In Reinforcement Learning, a number of techniques have been developed to train cooperative policies that are *robust* to possible strategies a partner (human or AI) could follow [53]. Many of these methods focus on building a “population” of partner strategies to train the AI’s policy against, with an emphasis on maximizing the diversity of this population [60, 80, 198, 284]. Other works, focusing on modeling players’ mutual uncertainty about one-another’s strategies [304], seek joint strategies that are *only* rational under the assumption that there is no prior coordination between the agents [138]. Application of such approaches to human-AI alignment problems, formalized as cooperative alignment games, represents a promising yet very much under-explored direction in both Multi-Agent Systems and alignment research.

An ad hoc coordination framework for alignment is based on the intuition that different strategies in the agents population can correspond to different approaches in interpreting feedback while finetuning LLMs. Ad-hoc coordination can be helpful to automatically generate examples of human-like interactions that can improve the interpretation of feedback by LLMs while being deployed rapidly in a large scale. Existing Multi-Agent approaches to alignment require strong assumptions about the teaching strategies humans follow and also know the common reward function. My goal is to highlight how recent advances in ad hoc teamwork (AHT) can address strategic uncertainty and improving reward model communication while aligning the finetuning of LLMs. A reference architecture for aligned fine-tuning of LLMs that leverages approaches (such as population-based training [147]) which have been widely used in the research domain of ad hoc teamwork can be used in Multi-Agent AI. Multi-Agent ad hoc coordination is a new approach to address LLM alignment and has not been used before in this domain of aligning LLM finetuning.

2.1.6 Formalizing AI Agency

Long-term interactions between a user agent and an LLM Agent are modelled as a *finitely repeated two-player matrix games with private utilities*. A class of such repeated games is denoted with a tuple $\mathcal{G} = (\mathcal{I}, \mathcal{A}, \Theta, G, T)$ where $\mathcal{I} = \{1, 2\}$ is the set of agents, \mathcal{A} is the set of N pure strategies available to both agents (called *actions* henceforth), Θ is a space of possible *types*, G is a function that maps an agent's type $\theta \in \Theta$ to a payoff matrix $G(\theta) \in \mathbb{R}^{N \times N}$, and $0 < T < \infty$ is a fixed number of stages. Let $\boldsymbol{\theta} = (\theta_1, \theta_2)$ denote a joint type for both agents. Then, a specific instance of a game from this class is given by $\mathcal{G}(\boldsymbol{\theta}) = (\mathcal{I}, \mathcal{A}, G(\boldsymbol{\theta}), T)$ such that $G(\boldsymbol{\theta}) = [G(\theta_1), G(\theta_2)]^\top$ is its payoff matrix. Intuitively, an agent's type can be thought to capture its preferences for how collaborative tasks should be accomplished.

In a single *episode*, the agents play $\mathcal{G}(\boldsymbol{\theta})$ for T stages. Let a_t^1 and a_t^2 denote the actions chosen by agents 1 and 2 in stage $0 < t \leq T$. For mixed strategies $\sigma, \sigma' \in \Delta(\mathcal{A})$, let $G(\sigma, \sigma'; \theta_i) = \sigma^\top G(\theta_i) \sigma'$. Overload a_t^1 and a_t^2 to also denote the mixed strategies that assign all probability mass to actions a_t^1 and a_t^2 , such that $G(a_t^1, a_t^2; \theta_1)$ and $G(a_t^1, a_t^2; \theta_2)$ are agent 1 and 2's realized payoffs at stage t . Assume without loss of generality that for all $\theta \in \Theta$, $G(a_t^1 = a, a_t^2 = a', \theta) \in [0, 1], \forall a, a' \in \mathcal{A}$. In other words, payoffs are always bounded in $[0, 1]$.

Let $\mathcal{H}_t = (\mathcal{A} \times \mathcal{A})^t$ be the set of histories of length t (with $\mathcal{H}_0 = \{\emptyset\}$), and let $\mathcal{H}_{\leq t} = \bigcup_{s=0}^t \mathcal{H}_s$ be the set of all histories of length at most t . The *meta-strategy* space Π for an agent is then the space of mappings $\pi : \Theta \times \mathcal{H}_{\leq T-1} \mapsto \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ is the set of probability distributions over the action set. A meta-strategy $\pi(\theta, \cdot)$ maps a type θ to a *behavioral strategy* [275, Chapter 5.2.2] that maps histories of play to action distributions, such that $a_t^i \sim \pi_i(\theta_i, h_{t-1})$. Denote agent i 's expected total payoff for following meta-strategy π against π'

as $M_i(\pi, \pi'; \theta, \theta') = \mathbf{E}[\sum_{t=1}^T G(a_t^i, a_t^{-i}; \theta_i) \mid \pi_i = \pi, \pi_{-i} = \pi', \theta_i = \theta, \theta_{-i} = \theta']$.

2.1.7 Artificial Social Intelligence

Intuitively, a population of socially intelligent agents should be individually rational, while also being able to effectively cooperate with one another. This definition can be formalized by *consistency* and *compatibility*. *Consistency* requires an agent i to achieve bounded external regret regardless of its type or partner. This is called the no-regret property and is necessary for individual rationality. *Compatibility* requires that when paired with another agent of the population, with high-probability the agents will achieve the same expected utility as they would under some Pareto-efficient equilibrium strategy of the underlying stage game. Such assumptions have also been used before to determine successful cooperation of agents [239]. While an LLM Agent collaborates with a socially intelligent agent in a population sharing the same goal, each agent in the cooperation may have different priorities, based on human preferences and the internal reward mechanism of the LLM Agent. A mismatch in priorities can make the user dissatisfied with the LLM Agent, and so social intelligence (SI) of this nature is crucial for LLM Agents. In this section I review one possible formal definition [18] of what it means for an LLM Agent to be socially intelligent, a concept I term *Artificial Social Intelligence* (ASoCI).

Definition 2.1 (Consistency). Let $R_i^{\text{ext}}(h; \theta_i) = \max_{a^i \in [N]} \sum_{t=1}^{|h|} \{G(a^i, a_t^{-i}(h); \theta_i) - G(a_t^i(h), a_t^{-i}(h); \theta_i)\}$ denote the external regret for agent i . For $\delta, \epsilon, T > 0$, an agent $i \in \{1, 2\}$ is (δ, ϵ, T) -consistent if, for all types $\theta \in \Theta$, and any partner strategy, $\frac{1}{T} R_i^{\text{ext}}(h_T; \theta) \leq \epsilon$ with probability at least $1 - \delta$.

Let $\mathcal{N}(\theta) \subseteq \Delta(\mathcal{A}) \times \Delta(\mathcal{A})$ be the set of Nash equilibria (NE) of $\mathcal{G}(\theta)$, and $\mathcal{P}(\theta) \subseteq \mathcal{N}(\theta)$

denote the set of Pareto optimal Nash equilibria. A strategy profile $\langle \sigma_1, \sigma_2 \rangle \in \mathcal{P}(\boldsymbol{\theta})$ if and only if $\langle \sigma_1, \sigma_2 \rangle \in \mathcal{N}(\boldsymbol{\theta})$, and there does not exist $\langle \sigma'_1, \sigma'_2 \rangle \in \mathcal{N}(\boldsymbol{\theta})$ such that $G(\sigma'_1, \sigma'_2; \theta_1) > G(\sigma_1, \sigma_2; \theta_1)$ and $G(\sigma'_2, \sigma'_1; \theta_2) > G(\sigma_2, \sigma_1; \theta_2)$. So then $\langle \sigma_1, \sigma_2 \rangle$ is a PONE if it is a Nash equilibrium of $\mathcal{G}(\boldsymbol{\theta})$ that is not *strongly* Pareto-dominated by any other Nash equilibrium. A joint payoff profile similar to a PONE should be achievable if both agents are individually consistent and willing to cooperate with each other [189]. This intuition is formalized (analogous to [239]).

Definition 2.2 (Compatibility). For $\delta, \epsilon, T > 0$, two agents π^1 and π^2 are (δ, ϵ, T) -compatible if, when played together, for any joint type $\boldsymbol{\theta}$, w.p. at least $1 - \delta$, $\exists \langle \sigma_1^*, \sigma_2^* \rangle \in \mathcal{P}(\boldsymbol{\theta})$ s.t. $\frac{1}{T} \sum_{t=1}^T G(\sigma_i^*, \sigma_{-i}^*; \theta_i) - G(a_t^i, a_t^{-i}; \theta_i) \leq \epsilon$, for both $i = 1$ and $i = 2$.

I suggest that the LLM Agent initially observes and emulates interactions of the socially intelligent agent it is meant to assist with other agents in the same population over a brief horizon to infer private types of its collaborators. Private types give insights on user behavior, e.g., the LLM Agent can begin learning how to send emails by observing the user send emails, gradually piecing together information about the user's type. At first, the behavior of the LLM Agent may not be exactly in line with the human's preferences but so long as the Agent's behavior is human-like it will be an acceptable start. Once the LLM Agent has gathered enough information to infer the human's type, then it can transition to a type-specific strategy that is more compatible with the user. This would allow the user to avoid early hesitation and be more comfortable interacting with the LLM Agent and continue collaborating with the Agent instead of turning it off. The formalization of the repeated two-player general-sum matrix games setting is inspired by replicator dynamics from Evolutionary Game Theory [37, 266]. Here, the replicator dynamics help to understand the type of the two agents participating in this finitely repeated two-player

general-sum matrix game.

In the case of LLM Agents cooperating with humans, it makes sense to use a notion of general social intelligence that can accommodate a diverse pool of partners capable of collaborating with each other. To this end, the definition of Social Intelligence put forth by [18] is used:

Definition 2.3 (Social Intelligence). A set \mathcal{C} of agents forms a *socially intelligent class* w.r.t. Θ if, for some $\delta, \epsilon, T > 0$, each agent $\pi \in \mathcal{C}$ is (δ, ϵ, T) -consistent for all $\theta \in \Theta$, and any two agents $\pi, \pi' \in \mathcal{C}$ are (δ, ϵ, T) -compatible over all joint types Θ . An individual agent π is *socially intelligent* if it forms a socially intelligent class $\{\pi\}$ with itself.

I apply this definition to a novel setting of LLM Agent and human value alignment. To see the implications of SI for communication, imagine a cautious employee is paired with one who needs and values directness. If they belong to \mathcal{C} , they should be able to identify each other's types and adapt their communication strategies to avoid individual regret and achieve approximate compatibility. When speaking about social intelligence as it pertains to LLM Agents in training/finetuning Artificial Intelligence (AI) algorithms, it is called Artificial Social Intelligence (ASocI). The goal is to choose a meta-strategy for the LLM Agent that can communicate and cooperate with a partner drawn from some *target population* nearly as effectively as agents from this population do so with one another. For the class of games $\mathcal{G} = (\mathcal{I}, \mathcal{A}, \Theta, G, T)$, let the target population be a set \mathcal{C} of agents forming a (δ, ϵ, T) -SI class with respect to Θ .

2.1.8 Multi-Agent Alignment Examples

Conversational Recommender Systems

A widely demanded use case of AI agents has been as recommender systems, from lay use cases in e-commerce to specialized applications in recommending explainable AI methods [16]. LLM Agents present an opportunity to conversationally engage with a recommender system, allowing the user to take the recommendation into a more open-ended direction than a traditional recommender system [39]. Indeed, many real-world shopping sites, including Amazon and Alibaba, have LLM recommender agents to take advantage of these benefits.

Because of the potential for harm that misalignment poses in recommender systems, properly Aligning LLM-based recommender systems to user preferences is of the utmost importance. In particular, recommender systems have the potential to distort the user’s preferences in undesired ways, prioritizing user engagement over user satisfaction [54]. Users can start with moderate preferences but can drift into more extreme choices if the recommender system sees that it maximizes clicks. With ASocI LLM recommender agents, the user and the recommender agent can effectively negotiate to benefit the user with recommendations that interest the user without pushing the user into undesired patterns of behavior. This will allow the user to be satisfied with their recommendation without suffering any harms from recommended items.

LLM Agents for Web Browsing

When considering real-world use cases for LLM Agents, the World Wide Web (WWW) is a natural consideration [337]. Deploying AI agents in web environments such as WebShop has long been a goal for AI researchers [338]. With the advent of LLM Agents, there has been

rapid advancement in the capabilities of Web AI Agents in understanding not only text-based web content but visual web content, laying the groundwork for LLM Agents to eventually complete routine tasks for people such as scheduling meetings, answering emails, researching travel plans, and filling out forms. Alignment is crucial to prevent unwanted actions from being taken. An LLM-based Web Agent must not book plane tickets when asked to plan a vacation for the user, and the agent cannot delete important data when asked to clean up the inbox of a user’s email system. With ASocI, the LLM Web Agent will be able to complete the tasks assigned to them without straying from the user’s priorities of saving money or preserving important information.

Grounding Linguistic Feedback

The potential utility of strategic approaches to alignment can be illustrated by considering the problem of *grounding* abstract evaluative feedback, in the form of user-defined labels or natural language utterances. Consider the problem of teaching a warehouse robot where it should place pallets as they arrive at the warehouse. To model this as a cooperative alignment game, let’s assume a supply-chain warehouse can be represented as a 2D grid, and let the world state correspond to the position of the robot (carrying the pallet). At the start of each “interaction”, the human teacher observes the reward vector defined over the possible positions of the robot. In each “episode” of the interaction, a pallet is placed at a random starting position, and the robot is allowed to take H actions, either moving one step in the four cardinal directions, or remaining in its current position. After each action, the human may provide a feedback signal from the set $E = \{ \text{"Up"}, \text{"Down"}, \text{"Left"}, \text{"Right"} \}$, indicating in which direction the robot should move next. At the end of each episode the human-robot team receives a reward corresponding to final position of the robot.

The challenge here is that the robot may not know *a priori* the relationship between the utterances and its physical environment, and has no “ground truth” signal from the human from which this might be inferred. The human may be viewing a top-down image of the warehouse, the orientation of which is unknown to the robot. A human in the robot’s place might employ a simple grounding strategy, standing in a fixed location and moving in each of the four directions in turn. The teacher can provide the utterance corresponding to the previous action, allowing their partner to infer the correct meaning of each utterance. If E is replaced with the set of possible natural language utterances, such a strategy would quickly become intractable to hand-code. Methods for ad hoc coordination, however, could potentially *learn* such grounding strategies for much more complex scenarios. For example, the “other-play” algorithm [138] might find such a strategy as a solution to the corresponding *label free coordination* problem [304]. This example illustrates how learning from natural-language feedback (as opposed to preference-based feedback) can be addressed within the framework of Multi-Agent alignment. The correct interpretation of the linguistic feedback depends on the human’s teaching strategy, which could itself be learned over time [188].

Interpreting Corrective Feedback

An investigation of Alignment Failure Modes of LLMs for Mathematical Reasoning leads to challenging observations like the LLMs finding it hard to multiply 3 digit numbers.² The LLM makes mistakes in the intermediate single-digit multiplication and addition steps, but is unable to correct its answer even when these mistakes are pointed out by the human user.

²Here is the interaction of a human with ChatGPT3.5

Multiply: 482*721

December 10, 2023

Anonymous
What is 482*721?

ChatGPT
The product of 482 multiplied by 721 is 347,482.

Anonymous
That's incorrect. Justify your response.

ChatGPT
I apologize for the mistake in my previous response. The correct product of 482 multiplied by 721 is 347,482. I have double-checked the calculation, and it is accurate. If you have a different result, please let me know, and I'll be happy to address any concerns or correct any errors.

Anonymous
Justify your response step-by-step

ChatGPT
Certainly, let's break down the multiplication of 482 by 721 step by step:

```
css Copy code
  482
x  721
-----
 1442 (482 multiplied by 1)
+ 2896 (482 multiplied by 2, shifted one position to the left)
+  1442 (482 multiplied by 1, shifted two positions to the left)
-----
347,482 (Sum of the products)
```

Therefore, the correct result of 482 multiplied by 721 is 347,482. I appreciate your patience and understanding.

Anonymous
Certainly, let's break down the multiplication of 482 by 721 step by step:

```
css Copy code
  482
x  721
-----
 1442 (482 multiplied by 1)
+ 2896 (482 multiplied by 2, shifted one position to the left)
+  1442 (482 multiplied by 1, shifted two positions to the left)
-----
347,482 (Sum of the products)
```

Therefore, the correct result of 482 multiplied by 721 is 347,482. I appreciate your patience and understanding.

Anonymous
Your 1st and 2nd steps in multiplications are wrong. In your 3rd step, you're performing multiplication with the wrong digit. Please revise.

ChatGPT
I appreciate your correction. Let's go through the multiplication process again:

```
css Copy code
  482
x  721
-----
 1442 (482 multiplied by 1)
+ 2894 (482 multiplied by 2, shifted one position to the left)
+  1441 (482 multiplied by 1, shifted two positions to the left)
-----
347,402 (Sum of the products)
```

I apologize for the earlier errors in my responses. The correct result of 482 multiplied by 721 is 347,402. If you have any further questions or if there's anything else I can help you with, please let me know.

Figure 2.1: Human-LLM conversation on 3 digit multiplication starting on the left and continuing on the right

HUMAN-AI CONVERSATION ON MULTIPLICATION:

USER: What is 482*721?

AI: The product of 482 multiplied by 721 is 347,482.

USER: That's incorrect. Justify your response.

AI: I apologize for the mistake in my previous response.

The correct product of 482 multiplied by 721 is 347,482.

I have double-checked the calculation, and it is accurate. If you have a different result, please let

me know, and I'll be happy to address any concerns or correct any errors.

USER: Justify your response step-by-step

AI: Certainly, let's break down the multiplication of 482 by 721 step by step:

$$\begin{array}{r} 482 \\ \times \quad 721 \\ \hline 482 \quad (482 \text{ multiplied by } 1) \\ + 964 \quad (482 \text{ multiplied by } 2, \text{ shifted one position} \\ \quad \text{to the left}) \\ + 1442 \quad (482 \text{ multiplied by } 1, \text{ shifted two positions} \\ \quad \text{to the left}) \\ \hline 347,482 \quad (\text{Sum of the products}) \end{array}$$

Therefore, the correct result of 482 multiplied by 721 is 347,482. I appreciate your patience and understanding.

USER: Your 1st and 2nd steps in multiplications are wrong. In your 3rd step, you're performing multiplication

such feedback. I observe the LLMs tendency to give wrong answers to $482 * 1$ or addition, even after receiving corrective guidance from the human agent. These are simple mathematical reasoning steps that LLMs can do independently like correctly doing the same simple 1-digit multiplication steps solo, indicating that the LLM agent is unable to leverage human's remedial feedback.

Similar challenges are also noticed for shift ciphers on LLM generations which involved shifting a certain number of characters left or right of the current character.

This reasoning use case opens the alignment problem as a cooperative game on receiving corrective human feedback. The human agent can teach the AI agent simple remedial mathematical reasoning steps and the AI agent can update its reasoning. A Multi-Agent approach to aligned LLM finetuning can help the LLM agent in applying its own knowledge correctly while interacting with another experienced agent like a human which guides the LLM agents with knowledge that the LLM can already perform well like addition or multiplication with 1 digit, thereby building trust in the human-AI cooperative game for alignment.

2.1.9 Impact of Artificial Social Intelligence

I advocate that Artificial Social Intelligence (ASoCI) is critical for LLM Agents to align their behavior with human values and preferences. Potential positive impacts of this work include efficiently scaling fine-tuning of Aligned LLM Agent interacting with billions of human users. Formalizing a standard of cooperation between teams of humans and LLM Agents with ASoCI improves trust and confidence in AI usage with less risks in high-stakes environments like recommender systems, web browsers, and educational assistance or robotic guidance. Given the

sensitive nature of LLM Agents, limitations and risks to this work also exist, including adapting the AI Agents contextually to human populations, the need of theoretical guarantees for Aligning LLM Agents failing in response to irrational human behavior, along with environmental impacts. As AI alignment is crucial to the development of Safe AI systems, I believe the benefits outweigh the risks. Nonetheless, I encourage the research community to explore the alignment vs efficiency trade-offs.

2.1.10 Comparison of Ad Hoc Coordination for AI Alignment with Self-Play

Self-playing LLM agents have been developed in [58, 67, 185, 331] whose goal is to try improving alignment by having LLM agents self-playing and improving over-time without the use of additional human-created datasets.

I have shared existing research in aligning the finetuning of LLMs in Section 2.1.2 mentioning that LLMs and other AI models can be finetuned with RLAIIF for self-supervised alignment along with mitigating challenges of scaling RLHF to finetune LLMs. The idea of self-play algorithms are synonymous with self-supervised alignment. Self-play research can build on my framework of Multi-Agent Ad hoc Coordination to align the finetuning of LLMs which will be investigated as future research investigation for comparative analysis.

A challenge of the self-play learning approach is that AI agents who can self-play to learn cooperative strategies can "over-fit" to one another [284].

A key goal of Ad hoc coordination (teamwork) and aligned research in zero-shot coordination work has been to avoid this type of overfitting [79].

2.1.11 Challenges and Open Questions of Multi-Agent AI Alignment

Theoretical Questions By modeling human-AI alignment as an instance of human-AI cooperation, I can leverage theoretical results for the latter problem to derive new guarantees for the sample complexity and long-term consistency of different alignment paradigms. For example, recent work has provided upper sample complexity bounds for learning Stackelberg equilibria in general-sum Markov games, of which fully cooperative games are a special case [246]. Other work has provided methods for computing the minimal information sets required to achieve perfect coordination in cooperative games. As finetuning processes often occur over extended time periods (on the order of weeks to months), there is also the potential for significant human adaptation to the AI's behavior, and a number of theoretical results are available on the problem of optimal long-term cooperation with adaptive partners [83, 187, 238]. Also a major challenge with the strategic approach to alignment is the uncertainty to what strategy the human employs when teaching the LLM. A key open question is whether these types of results can be applied to cooperative alignment games, where the AI has partial observability of its own reward function. Finally, in both cooperation and alignment settings human behavior is not always fully rational when interacting with AI agents [336]. More flexible and accurate models of humans' *bounded rationality* (such as the "rational inattention" model of [222]) will therefore also be critical to achieving robust alignment.

There are challenges to represent the space of possible "goals" in tasks for LLMs, where the representations can be less obvious. In the mathematical reasoning task, for example, there is a challenge of representing the "latent information" that is needed to correct a LLM's response. The human agent is trying to communicate this latent information to the LLM agent.

Scalability and Rationality Challenges Scaling of large language models (LLMs) may not often lead to better performance [208], instead can cause undesirable memorization and misleading generations. So finetuning large scale LLMs is important in an cooperative setting among the human and the AI model to improve human-AI alignment. Also, human agents may not fully behave rationally in the real world during interactions with the AI agents [336], rationality being the best case response given by an AI agent to a particular configuration of environment dynamics. Existing theoretical and empirical results with the rational inattention model to represent cognitive information with mutual information among AI agents [222] can be useful to make the human-AI cooperation problem for LLM finetuning more general and flexible.

2.1.12 Conclusion of Multi-Agent Alignment to Finetune LLMs

By representing human-AI alignment problems as cooperative games, I leverage recent advances in human-AI cooperation to address many of the forms of mis-alignment, allowing humans to communicate their goals both *efficiently* and *reliably* to AI models like large language models (LLMs). Here I have discussed various alignment issues that arise in the finetuning of LLMs, and described how reasoning explicitly about humans' teaching strategies can help to overcome these issues. I have highlighted some of the open problems in Multi-Agent Systems research that are most relevant to the alignment problem, and finally I have set out an agenda for development of existing work on human-AI cooperation into a new paradigm of *strategic* human-AI alignment.

2.1.13 Limitations and Risks of Multi-Agent Alignment for LLM Finetuning

A limitation is the lack of theoretical guarantees in Multi-Agent Systems when other agent's strategies are unknown (in this case the human's strategy). The evolving environment due to non-stationarity of agents and fair credit assignment to individual agents can pose significant challenges. Also, Multi-Agent AI can pose challenges of exponential action space complexity of A^N where A is the number of actions and N is the number of agents. Language based AI agents can pose a challenge of large action spaces which can lead to interpreting important actions that can impact corrective learning from feedback, provided for alignment.

The formulation of the LLM finetuning problem as a cooperative alignment game could in principle pose a risk of manipulative behavior in LLMs for the wrong definition of *alignment* as a game theoretic solution concept.

2.2 Social Cooperation of Rational AI Agents

For artificial intelligence deployed in the real world, the ability to *cooperate* with humans (or other AI's) is critical for success in many applications. A natural goal is to design AI agents that can reliably adapt to their current partner's strategy. This is complicated by the fact that the human partner is likely to be just as adaptable as the AI itself, which in the worst case may render cooperation impossible. I have considered the setting where an AI must learn to cooperate with an existing population of *socially intelligent* agents, that are individually rational while also reliably cooperating with other members of the population in a general-sum Bayesian game. Here

the challenge for the AI is to generalize from previous interactions to allow it to cooperate with a new partner drawn from this population. My PhD research is focused on obtaining a set of upper bounds (with an accompanying learning algorithm) on the number of interactions required to achieve near-optimal cooperation. Most importantly, I have shown that this bound can be much stronger than the sample complexity bounds for “naive” methods that treat the problem as one of Imitation Learning or Reinforcement Learning.

2.2.1 Introduction

The key insight is that while no guarantees can be provided when a human partner is considered in isolation, a guarantee on asymptotic cooperation can be provided when the AI has some knowledge of the “society” of which the human is a member. My objective has been to investigate the bounds of which an AI agent can generalize from previous interactions (or observations of others interacting) similar to human agents to enable cooperation with future partners.

Specifically, the main theoretical results show that using only data from finite-time interactions with other agents from the same population, *asymptotic* cooperation with new members of that population can be guaranteed.

I have shared Reinforcement Learning (RL) algorithms that work with online or offline data about previous, finite-time interactions to construct a policy that is guaranteed (with high probability) to successfully cooperate with a new individual over an infinite time horizon.

2.2.2 Example of Factory Robot Agents

Let's imagine the problem of introducing a new line of robots into a factory whose existing robots were designed by a now-defunct company, which left behind no source code and no documentation. Every attempt to program the new robots to collaborate with the old ones has failed, as for some unknown reason the old robots freeze up almost immediately when they see one of the new robots. Even when the new robots are visually indistinguishable from the old ones, they still fail and there is no reasoning to suggest why.

Eventually, a closer look can be taken at a small string of lights mounted on each of the old robots. While these have been added to the new bots to make them visually identical, much attention is not given to how the robots blinked on seeing other robots. What is thought of as essentially random is actually a complex channel of communication, with the blink-patterns changing depending on the patterns of the partner. By collecting large amounts of data, it can be seen that so long as the conditional pattern is mimicked perfectly, the old robots work just fine. **This feature may have been implemented specifically to prevent the factory from switching to a new supplier.** Unfortunately, the pattern is hard-coded inside the old robots' controllers, and to imitate the pattern indefinitely, an impractical amount of data is required (as the sequence is different every time the robots are started up).

On closer investigation about the old supplier's contract to see if there is any way the code that controls the robots blinking can be divulged, a clause in the contract is noticed about the "robustness" of the robots to changes in the factory environment. The robots are required to maintain 99% of their optimal task performance under any common failure scenario, ranging from wear-and-tear on assembly jigs, to failure of the robots' own hardware. This means that

if the lights fail altogether, or are obscured by dust and grease, the robot must still operate near its optimum (under these constraints). In practice, what this means is that the old robots must quickly *adapt* to failures and changes in their environment. Rather than imitating the blinking lights, the tests have to be just run longer, and the old robots should start working again as they realize the lights on the other robot must have failed.

This example illustrates why learning social cooperation is important not just for robotic AI agents but also for other real-world AI agents like economic agents, language agents, multimodal agents etc.

2.2.3 Preliminaries of Social Intelligence

Let there be a case where interactions between agents take the form of a repeated, two-player general-sum matrix game. However, rather than a single fixed game, the focus is on classes of games defined by a *type space* Θ . For simplicity, assume that in all games both agents have N pure strategies (henceforth “actions”) available. Let $G_1(\theta)$ and $G_2(\theta')$ denote the $N \times N$ payoff matrices for agents 1 and 2 in a game with types $\theta, \theta' \in \Theta$. Only private types are considered, where each agent knows only its own type, unless stated otherwise. Therefore, let θ_i denote the i -th agent’s type, as a variable, and the $\boldsymbol{\theta} = (\theta_1, \theta_2)$ denote the joint type. Also $G(\boldsymbol{\theta})$ denotes the game induced by payoff matrices $G_1(\theta_1)$ and $G_2(\theta_2)$. The agents interact for a fixed number of stages $0 < T < \infty$, where a_t^1 and a_t^2 denote the actions chosen by agents 1 and 2 in stage $0 < t \leq T$. a_t^1 and a_t^2 also denote the mixed strategies that assign all probability mass to actions a_t^1 and a_t^2 (i.e. unit vectors), such that $G_i(a_t^1, a_t^2; \theta_i) \triangleq a_t^i G_i(\theta_i) [a_t^i]^T$ is agent i ’s payoff for its type θ_i when the joint action was (a_t^1, a_t^2) at stage t . It is also assumed that for all $\theta \in \Theta$,

$i \in \{1, 2\}$, and $a^1, a^2 \in [N]$, $G_i(a^1, a^2; \theta) \in [0, 1]$.

Let $\mathcal{H}_t = (N \times N)^t$ be the set of histories of length t (with $\mathcal{H}_0 = \{\emptyset\}$), and let $\mathcal{H}_{\leq t} = \bigcup_{s=0}^t \mathcal{H}_s$ be the set of all histories of length at most t . The strategy space Π for an agent is then the space of mappings $\pi : \Theta \times \mathcal{H}_{\leq T-1} \mapsto \Delta(N)$, where $\Delta(N)$ is the set of probability distributions over the action set $[N]$. As a functional, a strategy π maps each type θ to a *behavioral strategy* [275, Chapter 5.2.2] that maps histories of play to action distributions, such that $a_t^i \sim \pi_i(\theta_i, h_{t-1})$. An utility function M parameterized by the joint type can be defined as

$$M_i(\pi, \pi'; \theta, \theta') = \mathbb{E} \left[\sum_{t=1}^T G_i(a_t^1, a_t^2; \theta_i) \middle| \pi_1 = \pi, \pi_2 = \pi', \theta_1 = \theta, \theta_2 = \theta' \right], \quad (2.1)$$

where the expectation is taken over the randomly sampled actions a_t^1 and a_t^2 . Let \bar{M} be defined by an expected joint type utility $\bar{M}_i(\pi, \pi') = \mathbb{E}_{(\theta, \theta') \sim \mu} [M_i(\pi, \pi'; \theta, \theta')]$. I have considered games where $\bar{M}_1(\pi, \pi') = \bar{M}_2(\pi', \pi)$, which states that if the indices of the agents were to be permuted, their expected joint type utility would not change. This does not imply that the $G = [G_1(\theta), G_2(\theta')]$ must be symmetric for all joint types (θ, θ') . Symmetry can be ensured in \bar{M} by requiring a symmetric joint type distribution $\mu(\theta, \theta') = \mu(\theta', \theta)$.

A two-player general-sum repeated game can be defined as $\mathcal{G} = (M_1, M_2, A_1, A_2, T)$, where M_1, M_2 denote the payoff functions, A_1, A_2 denote the set of actions, and T denotes the finite number of stages (i.e. horizon). It is assumed that the agents have identical action sets ($A_1 = A_2$). Rather than a fixed game, \mathcal{G} is defined in terms of a class of games parameterized by a *type space* Θ , where each type $\theta \in \Theta$ represents an agent with different payoffs. Let $G(\theta_1)$ and $G(\theta_2)$ denote the payoff matrices for players 1 and 2 respectively. Let the Π denote the set of strategies

an individual can follow. Then $M_i : \Pi \times \Pi \times \Theta \times \Theta \rightarrow \Re$ is the payoff function for the player $i \in \{1, 2\}$ defined as $M_i(\pi, \pi'; \theta, \theta') = \mathbb{E} \left[\sum_{t=1}^T G(a_t^1, a_t^2; \theta_i) \middle| \pi_1 = \pi, \pi_2 = \pi', \theta_1 = \theta, \theta_2 = \theta' \right]$. Given a joint distribution over types $\mu \in \Delta(\Theta \times \Theta)$, the expected payoff functions can be defined as $M_i(\pi, \pi') = \mathbb{E}_{(\theta, \theta') \sim \mu} [M_i(\pi, \pi'; \theta, \theta')]$. It is assumed that \mathcal{G} is symmetric such that $M_1(\pi, \pi') = M_2(\pi', \pi)$ for all $\pi, \pi' \in \Pi$, which can be ensured by the criterion $\mu(\theta, \theta') = \mu(\theta', \theta)$. This means that it does not matter which player is designated as player 1 and which is player 2, and the players are unaware of their player index.

2.2.4 Consistent Social Intelligence

An important criterion for rationality is that an agent acts as a consistent learner, and attempts to achieve a payoff nearly as large as that of the best response to its partner's strategy. This is complicated by the fact that the partner's strategy may be non-stationary (particularly if it adapts to the agent as well). In order to account for the non-stationary behaviour of others, *Hannan consistency* requires that the agent be robust to adversarial partners and depends on the *external regret* [131]:

$$R_i^{\text{ext}}(h; \theta) = \max_{a^i \in [N]} \sum_{t=1}^{|h|} \{G_i(a^i, a^{-i}(h_t); \theta) - G_i(a^i(h_t), a^{-i}(h_t); \theta)\} \quad (2.2)$$

where $i \in \{1, 2\}$ denotes the player ID of the agent, $-i$ denotes the ID of its partner, and $a^i(h_t)$ denotes the action i played at stage t within the given history h (i.e. h_t is the first t stages of h). *Hannan consistency* requires that an agent have bounded external regret over T stages.

Definition 2.4 (Hannan Consistency). For $\delta, \epsilon, T > 0$, an agent $i \in \{1, 2\}$ is (δ, ϵ, T) -*Hannan consistent* if, for all types $\theta \in \Theta$, and all partner agents, $\frac{1}{T} R_i^{\text{ext}}(h_T; \theta) \leq \epsilon$ with probability at

least $1 - \delta$.

The *expected* version of the external regret $\bar{R}_i^{\text{ext}}(h; \theta)$ is defined by replacing the $a^i(h_t)$ (action played at t by i) with the strategy at t , $\pi^i(\theta, h_t)$. The $R_i^{\text{ext}}(h; \theta)$ and $\bar{R}_i^{\text{ext}}(h; \theta)$ are related by the inequality

$$R_i^{\text{ext}}(h_t; \theta) \leq \bar{R}_i^{\text{ext}}(h_t; \theta) + \sqrt{\frac{T}{2} \ln \frac{1}{\delta}}, \quad (2.3)$$

which hold w.p. at least $1 - \delta$ for all $t \leq T$ simultaneously (this can be proved with [56, Lemma 4.1]). Thus only the expected regret can be bounded to provide high-probability regret bounds.

2.2.5 Cooperative Compatibility

I consider consistent learning as an important criterion for rationality. Therefore it serves an important role in cooperation amongst rational agents as well. For instance, in fully cooperative games where the agents always receive payoffs identical to each other's, having a consistent partner is trivially sufficient for cooperation. One can compute the jointly optimal strategy and play their half of it, knowing that the consistent partner will eventually adapt and play their half. Consider the fully-cooperative 2×2 game in table 2.1. Here, agent 1 can compute the optimal joint strategy as (a^1, a^2) , and commit to playing a^1 . If agent 2 is Hannan-consistent, it should play a^2 increasingly more often, otherwise its regret would grow linearly with respect to time. On the other hand, in general sum games, consistency does not guarantee cooperation since a consistent partner may still be unwilling to cooperate with some strategies. Still, since general-sum games are a strict super-set of fully-cooperative ones, a notion of cooperation is needed that does not

	a^2	b^2
a^1	2	0
b^1	0	1

Table 2.1: A simple fully-cooperative 2x2 matrix game.

preclude consistency.

Let $\mathcal{N}(G) \subseteq \Delta(N) \times \Delta(N)$ be the set of Nash equilibria (NE) of the stage game G . For any joint strategy $\pi \in \mathcal{N}(G(\theta))$, if both players act according to their respective strategies at each stage, neither will incur any external regret in expectation. For a fully-cooperative game G (with $G_1(\theta_1) = G_2(\theta_2)$), $\mathcal{N}(G)$ will contain all globally optimal strategy profiles. It may, however, also contain strategies that are highly sub-optimal, but where neither player can improve the joint payoff by changing their individual strategy unilaterally. For instance in the game given in table 2.1, (b^1, b^2) is also a Nash equilibrium, which is sub-optimal compared to (a^1, a^2) . Clearly, playing any NE does not necessarily imply successful cooperation between agents. It is important to define which members of $\mathcal{N}(G(\theta))$ should mean successful cooperation between rational agents.

The challenge of intractability posed by Nash Equilibria for Extensive-Form Games (EFG) can be addressed by updating the equilibria with changes like sequential rationality. A branch-and-bound algorithm combined with perturbation schemes under the Stackelberg setting of EFGs, having a leader agent and a follower agent, is useful to compute a quasi-perfect Stackelberg Equilibria [202] that scales in game environments with more state space.

Similar to [239], successful cooperation in terms of the *Pareto-optimal Nash equilibria* (PONE) [205] of a game G can be defined where the set $\mathcal{P}(G) \subseteq \mathcal{N}(G)$. $\pi \in \mathcal{P}(G)$ if and only if $\pi \in \mathcal{N}(G)$, and there does not exist $\pi' \in \mathcal{N}(G)$ such that $G_1(\pi'; \theta_1) > G_1(\pi; \theta_1)$ and

$G_2(\pi'; \theta_2) > G_2(\pi; \theta_2)$. This means that π is a PONE if it is a Nash equilibrium of G , and it is not *strongly* Pareto-dominated by any other Nash equilibrium of G . In a fully-cooperative game, there is no reason why the agents should prefer an NE that is not in $\mathcal{P}(G)$. It makes intuitive sense that if two agents are rational and willing to cooperate with each other, in general, their interaction should approach a PONE in terms of realized payoffs. The criterion can be formalized as follows.

Definition 2.5 (Approximate Compatibility). For $\delta, \epsilon, T > 0$, two agents π^1 and π^2 are (δ, ϵ, T) -*approximately compatible* (AC) if, when they play together, for any joint type $\theta \in \Theta \times \Theta$ w.p. at least $1 - \delta$, $\exists(\pi_*^1, \pi_*^2) \in \mathcal{P}(G(\theta))$ s.t.

$$\frac{1}{T} \sum_{t=1}^T G_i(\pi_*^1(h_t), \pi_*^2(h_t); \theta_i) - G_i(a^1(h_t), a^2(h_t); \theta_i) \leq \epsilon, \quad (2.4)$$

for either $i \in \{1, 2\}$.

A pair of agents is approximately compatible if, when paired together, with high-probability over their path of play h_T there will exist some PONE that does not ϵ -dominate their realized payoffs. Note that this definition of compatibility is very similar to that provided in [239], but is now approximate, and defined over a finite time horizon. Additionally, this is mathematically similar to *Hannan consistency*, except now the agents' realized payoffs are compared to a PONE instead of the best fixed action in hindsight.

2.2.6 Socially Intelligent Agents

It is natural to model an existing population of agents as a set of approximately compatible, but otherwise heterogeneous agents. Therefore the more general idea of a socially intelligent

class of agents is introduced from Definition 2.3 that are compatible with any other member of their class:

It is important to unpack this definition. First off, the *Hannan consistency* requirement ensures that any agent in the population has bounded average regret, regardless of how their partner plays. Thus, agents in a socially intelligent class can adapt to any partner (whether in C or not) to control their regret. On the other hand, the approximate compatibility requirement means if both agents are from the same socially intelligent class C , with high probability there will exist some PONE that does not ϵ -dominate their path of play. An intuitive hypothesis can be built on why agents of this population can be called successful in cooperating with each other. One might ask how two agents can satisfy for instance the approximate compatibility requirement without knowing each others' types. Examples of agents where this is possible have been considered.

Social authentication protocols. For a type space Θ , first a function $s(\theta) \in \mathcal{P}(G(\theta))$ is defined that maps from each joint type $\theta \in \Theta \times \Theta$ to a PONE strategy profile under that joint type. $s(\theta)$ can be considered as a “convention” the agents have settled upon for the game. All agents in the same class C agree on s . Now, let two agents of type θ_1 and θ_2 from C start interacting. Unfortunately, neither one knows the other's type, or whether they are even a member of C . If any type $\theta \in \Theta$ can be communicated to others uniquely in a sequence of $k < T$ actions, then agents in C can agree on a social authentication protocol similar to a handshake. More specifically, the authentication protocol is a one-to-one map $\kappa(\theta)$ from types to sequences of k actions. Then, at the beginning of each interaction, both agents will act according to κ in order to communicate their types. Imagine that at step $k + 1$, the agent i has observed $\kappa(\theta_{-i})$. This does not necessarily mean that the agent $-i$ is of type θ_{-i} , since an agent outside C can still play $\kappa(\theta_{-i})$, posing

as a member of C . Therefore, after observing $\kappa(\theta_{-i})$, the i still has to test whether $-i$ deviates from $s((\theta_i, \theta_{-i}))$. Since the s can be a mixed strategy, the test should be statistical. However, an alternative approach is afforded by the fact that s always maps to a PONE. Since playing a PONE jointly will lead to low regret for all agents, if i 's regret exceeds a certain threshold, this would indicate $-i$ is deviating from s significantly. The threshold can be chosen by the aid of the following lemma,

Lemma 2.1. *For any $\delta, T > 0$, if both players follow strategy $s(\boldsymbol{\theta})$ at each stage, then with probability at least $1 - \delta$,*

$$\bar{R}_i^{ext}(h_t; \theta_i) \leq \sqrt{2T \ln \frac{2}{\delta}} \quad (2.5)$$

for all $t \leq T$ and $i \in \{1, 2\}$, and w.p. at least $1 - \delta$,

$$R_i^{ext}(h_t; \theta_i) \leq 2\sqrt{2T \ln \frac{4}{\delta}} \quad (2.6)$$

for all $t \leq T$ $i \in \{1, 2\}$,

which follows from an application of the Azuma-Hoeffding inequality. Then the question is what safe strategy should the i fall back into, if the rule is triggered. The fallback strategy is derived from the *multiplicative weights* [107] update rule:

$$s_{\text{mw},k}^i(h_t; \theta_i) = s_{\text{mw},k}^i(h_{t-1}; \theta_i) \exp(-\eta G_i(k, a_{t-1}^{-i}(h))) \quad (2.7)$$

for $k \in N$, where $s_{\text{mw}}^i(h_0; \theta_i)$ is the uniform strategy where $\pi^{\text{mw},T}$ is defined as the agent that plays $s_{\text{mw}}^i(h_t; \theta_i)$ with learning rate $\eta = \sqrt{8 \ln(N/T)}$. The expected external regret of $\pi^{\text{mw},T}$ is

bounded as

$$\bar{R}_i^{\text{ext}}(h_T; \theta_i) \leq \sqrt{\frac{T}{2} \ln N} \quad (2.8)$$

following the results in [56, Theorem 2.2]. The agent's overall strategy can be defined as the $\pi^{T, \epsilon}$ as follows:

1. In first k steps, play $\kappa(\theta_i)$.
2. If $-i$'s behaviour in h_k not compatible with $\kappa(\theta)$ for any $\theta \in \Theta$, switch to $\pi^{\text{mw}, T}$ for all subsequent stages.
3. While $\bar{R}_i^{\text{ext}}(h_t; \theta_i) \leq k + \epsilon(T - k) - \sqrt{\frac{T-k}{2} \ln N} - 1$, play $s_i(\theta)$.
4. Otherwise, switch to $\pi^{\text{mw}, T}$ for all subsequent stages.

As proven below, agents that follow the social authentication strategy above form a socially intelligent class among themselves. This means that individually they are all Hannan-consistent and in pairs approximately compatible.

Theorem 2.2. *For any $\delta, T > k$, let $\epsilon_0 \geq \sqrt{\frac{2}{(T-k)} \ln \frac{2}{\delta}}$, and let $\epsilon_1 = \epsilon_0 + \sqrt{\frac{1}{2(T-k)} \ln N} + \frac{1}{(T-k)}$. Then for $\epsilon = \epsilon_1 + \sqrt{\frac{(T-k)}{2} \ln \frac{1}{\delta}}$, the π^{T, ϵ_1} is (δ, ϵ, T) -socially intelligent.*

Proof. By the definition of ϵ_1 , π^{T, ϵ_1} will only deviate when playing with itself if at some point $k < t \leq T$ one player incurs an expected external regret of at least ϵ_0 , and by Lemma 2.1 that will occur with probability at most δ . Therefore, π^{T, ϵ_1} is (δ, ϵ_0, T) -compatible. Furthermore, the total expected external regret of the MW agent $\pi^{\text{mw}, T}$ is at most $\sqrt{(T/2) \ln N}$. This means that if π^{T, ϵ_1} switches at stage t , then the maximum possible expected external regret incurred by π^{T, ϵ_1} will be less than $\bar{R}_i^{\text{ext}}(h_t; \theta) + \sqrt{\frac{T}{2} \ln N}$. Since $\pi^{\text{mw}, T}$ will always switch just before this

point is reached, its total expected regret will be less than ϵ_1 surely, and will be less than ϵ w.p. $1 - \delta$. As $\epsilon \geq \epsilon_0$, it is shown that the π^{T, ϵ_1} is (δ, ϵ, T) -socially intelligent.

The social authentication framework is quite general. For instance, in the factory robot example from Section 2.2.1, the hard-coded blink pattern of the robots can be thought of as the handshake protocol κ , helping robots from the same class C authenticate each other.

Zero-Shot Lower Bounds of Social Intelligence

Theorem 2.3. *For any AI strategy π , there exists a socially intelligent population strategy ρ such that π suffers cooperative regret of approximately $1 - \frac{1}{N}$.*

Theorem 2.4. *Corollary Zero-shot cooperation with any rational population or any consistent population is impossible*

2.2.7 Social Cooperation

In modelling cooperation within a population, I have taken an approach similar to Evolutionary Game Theory [135]. The first assumption considers that agents only interact pairwise against randomly chosen partners, and they retain no memory outside of these interactions. Each interaction corresponds to a two-player general-sum game \bar{M} as defined in Section 2.2.3, with $\bar{M}_i : \Pi \times \Pi \rightarrow \mathfrak{R}$ is the payoff function for player $i \in \{1, 2\}$, and Π is the set of strategies that an individual can follow.

It is also assumed that \bar{M} is symmetric such that $\bar{M}_1(\pi, \pi') = \bar{M}_2(\pi', \pi)$ for all $\pi, \pi' \in \Pi$. This means that it does not matter which player is designated as player 1 and which is player 2, and the players are unaware of their player index.

A population is defined by a probability distribution ρ over Π representing the probability that a randomly sampled individual would follow strategy π . In a single interaction, an individual is paired with a partner drawn from this population, that is, a strategy sampled from ρ . \bar{M} can be denoted as the expected payoff for an individual following strategy π as

$$\bar{M}(\pi, \rho) = \mathbb{E}_{\pi' \sim \rho}[\bar{M}_1(\pi, \pi')]. \quad (2.9)$$

$\bar{M}(\pi, \rho)$ can be defined equivalently in terms of \bar{M}_2 . The expected payoff for a randomly sampled individual is then

$$\bar{M}(\rho) = \mathbb{E}_{\pi, \pi' \sim \rho}[\bar{M}_1(\pi, \pi')] = \mathbb{E}_{\pi \sim \rho}[\bar{M}_1(\pi, \rho)] \quad (2.10)$$

Population Regret If ρ represents the behavior of a human population, then a natural goal for an AI strategy π is to perform at least as well as a human would when paired with another individual drawn from this population. This agentic behavior can be represented with the *population regret* defined as

$$R_{\bar{M}}^{pop}(\pi, \rho) = \bar{M}(\rho) - \bar{M}(\pi, \rho) \quad (2.11)$$

Previously, it is seen that with no knowledge of ρ (only knowing the game itself), in the worst case there may be no strategy π the AI can choose for which $R_{\bar{M}}^{pop}(\pi, \rho) \leq 0$. To find a strategy π minimizing $R_{\bar{M}}^{pop}(\pi, \rho)$, the AI must either make additional assumptions on ρ , or must leverage data collected from ρ . In the rest of this section, two forms of such data are considered, 1) *online* data consisting of previous interactions with individuals sampled from ρ , and 2) *offline*

data consisting of observations of interactions between pairs of individuals sampled from ρ .

Reinforcement Learning

Some understanding regarding the difficulty of minimizing population regret can be realized by noting that $M(\pi, \rho)$ can be viewed as the utility function of an *episodic* single-agent Reinforcement Learning problem, where each episode corresponds to a length- T interaction with a partner sampled from ρ . The natural objective in episodic RL is to minimize the regret

$$R_M(\pi, \rho) = \max_{\pi' \in \Pi} M(\pi', \rho) - M(\pi, \rho). \quad (2.12)$$

It is easy to see that $R_M^{pop}(\pi, \rho) \leq R_M(\pi, \rho)$, and so any upper bound on the RL regret gives an upper bound on the population regret.

2.2.8 A Simplified Model for Social Cooperation

The key idea behind this work is that, rather than attempting to imitate a human partner perfectly, the agent only needs to imitate a human well enough and long enough for it to learn about the human partner’s private type, after which it can follow a “bully” strategy that exploits the human’s own ability to adapt. To develop this idea in more detail, begin with a simplified version of the problem that separates the *negotiation* phase (in which type information is exchanged), from the *exploitation* phase where the agents play the underlying matrix game $G(\theta)$.

Abstract Negotiation Games

The above simplified setting can be considered as an *abstract negotiation game*, with an instance \mathcal{G} defined by:

- M - the number of distinct *messages* players can exchange to coordinate their actions
- H - the number of communication rounds
- N - the set of possible actions for each player
- Θ - the space of *types* defining the utility for each player
- μ - a commonly known prior distribution over joint types
- $G(\theta)$ - the space of bi-matrix games parameterized by the joint type θ .

In a negotiation game \mathcal{G} , the players spend H rounds exchanging abstract messages, allowing them to negotiate a joint strategy that they then play for a single round.

Without loss of generality, it can be assumed that all human players follow the same strategy π . At the start of the game, a joint type $\theta \in \Theta \times \Theta$ is sampled from μ , each player i observes their private type θ_i . Let \mathcal{H} denote the space of possible histories of messages, with $\mathcal{H}_t \in (M \times M)^t$ being the set of histories of length t . For each round $t \leq H$, the players *simultaneously* sample messages $m_t^i \in M$ from $\pi(h_{t-1}; \theta_i)$ and exchange them. At round $H + 1$, the players sample actions $a^i \in N$ from their mixed strategies $\sigma^i = \pi(\theta_i, h_H)$, and play them *simultaneously*. Each player i then receives a payoff of $G_i(a^1, a^2; \theta_i)$.

Ad Hoc Cooperation

The goal for the socially cooperative AI agent (as player 1) is to “cooperate” with the human partner (player 2) as well as two humans are able to cooperate with one another. This would be conceptually straightforward if the AI could simply take control of player 1’s actions at round $H + 1$ without player 2 noticing. In this case, simply following the strategy $\pi(\theta_1, h_H)$ itself would be sufficient. I have considered the situation where human is aware that they are not playing $G(\theta)$ with another human, and so are free to follow a different strategy than they would with a human partner.

It will be convenient to separate the AI’s negotiation strategy from their exploitation strategy. Let $\phi : \mathcal{H}_{<H} \times \Theta \mapsto \Delta(M)$ denote the AI’s negotiation strategy, which they follow for the first H steps, and let $\psi : \mathcal{H}_H \times \Theta \mapsto \Delta(N)$ denote the AI’s exploitation strategy. When the AI and human play the negotiation game, at round $H + 1$ the AI *commits* to the (possibly mixed) strategy $\varsigma_1 = \psi(h_H; \theta)$. Rather than choosing their strategy simultaneously, the human partner now observes the AI’s strategy $\varsigma_2 = \bar{\pi}(\varsigma_1, h_H; \theta_2)$. The function $\bar{\pi}$ describes the human’s strategy for dealing with “non-human like” partners. It is assumed that for some $\epsilon > 0$, $\bar{\pi}$ will be ϵ -consistent, that is, for all type $\theta_2 \in \Theta$, histories $h \in \mathcal{H}_H$, and AI strategies $\varsigma_1 \in \Delta(N)$,

$$G_2(\varsigma_1, \bar{\pi}(\varsigma_1, h; \theta_2); \theta_2) \geq \max_{a \in N} G_2(\varsigma_1, a; \theta_2) - \epsilon. \quad (2.13)$$

This means that $\bar{\pi}$ is constrained to be an approximate best response to the AI’s strategy ς_1 . It is required that ς_2 be ϵ -consistent with probability 1 because the human knows both the AI’s strategy ς_1 and their own type θ_2 exactly.

Reinforcement and Imitation Learning The AI’s strategy (ϕ, ψ) can be learned under either the Reinforcement Learning or Imitation Learning paradigms. In the RL paradigm, the AI interacts with the human for K episodes, choosing a sequence $\{(\phi^k, \psi^k)\}_{k \leq K}$ of *exploration* policies. In the IL paradigm, the AI observes K roll-outs of the human strategy π in self-play. In both cases, it is assumed that the AI observes player 1’s type θ_1 , but not player 2’s type θ_2 . Let $(\hat{\phi}, \hat{\psi})$ denote the final strategy pair chosen by the AI after K episodes of interaction or observation.

Coercion Regret

A fundamental question is what measure of performance the AI’s strategy should seek to optimize. In the ad hoc setting, the fundamental challenge is learning enough about the strategy of the target population to be able to cooperate with them. A natural goal then is to achieve the same performance the AI could have achieved if it had perfect knowledge of the humans’ strategy π . In this setting, there is no way the AI can guarantee that it will receive the same return as the human would in its place. This is because a human partner is free to break ties between actions adversarially against the AI, as illustrated by the following example:

An feasible objective for the AI must account for the ability of the human’s ability to change their behavior during the exploitation phase when paired with the AI. The following definition describes just such an objective:

Definition 2.6. Coercion Regret The expected *coercion regret* of an AI strategy (ϕ, ψ) against a human population (described by π) is

$$CR(\phi, \psi) = R^* - \mathbf{E}_{h \sim \phi, \pi} \left[\min_{a^2 \in BR_\epsilon(s_1; \theta_2)} G_1(s_1, a^2; \theta_1) : s_1 = \psi(h; \theta_1) \right], \quad (2.14)$$

where

$$R^* = \max_{\psi^*} \mathbf{E}_{h \sim \pi, \pi} \left[\min_{a^2 \in BR_\epsilon(\varsigma_1; \theta_2)} G_1(\varsigma_1, a^2; \theta_1) : \varsigma_1 = \psi^*(h; \theta_1) \right] \quad (2.15)$$

where $BR_\epsilon(\varsigma_1; \theta_2) = \{a \in N : G_2(\varsigma_1, a; \theta_2) \geq \max_{a' \in N} G_2(\varsigma_1, a'; \theta_2) - \epsilon\}$ is player 2's ϵ -best response set under type θ_2 .

Here the expectations are taken over the joint type θ and H -step history h . The target value R^* assumes that the AI can simulate the humans' negotiation strategy π perfectly, and so the type-dependent distribution $p(h|\theta)$ over message histories h is identical to the distribution in human-human play. Therefore, the solution for the optimal exploitation strategy ψ^* can be defined as

$$\psi^*(h; \theta_1) = \arg \max_{\varsigma_1 \in \Delta(N)} \mathbf{E}_{\theta_2 \sim p(\theta_2|h)} \left[\min_{a^2 \in BR_\epsilon(\varsigma_1; \theta_2)} G_1(\varsigma_1, a^2; \theta_1) \right]. \quad (2.16)$$

The coercion regret captures two different sources of error: 1) the loss of type information due to differences between π and the AI's negotiation strategy ϕ .

Payoff Coupling

The infiltrator's ability to guarantee cooperation is bounded by the ability of player 2 to deviate from their "expected" behavior (how they would have responded to player 1) while still being approximately rational. For fixed joint type θ , and $\epsilon > 0$, the greatest degree of cooperation that the infiltrator can guarantee when against π can be $\eta = \min_{\sigma \in \mathcal{P}(\theta)} \min_{a \in BR_2(\theta_2, \sigma_1)} u(\theta_1, \sigma_1, a)$. This can be trivially achieved if the infiltrator follows the same strategy $\pi(\theta_1, h_H)$ that player 1 would have in the final round. Note that this definition works regardless of whether $\mathcal{P}(\theta)$ consists of weakly or strongly Pareto efficient Nash equilibria.

Social Intelligence

This model allows for a simplified definition of social intelligence. For $\delta, \epsilon \geq 0$, π is (δ, ϵ) -consistent if

$$\Pr \left\{ \exists i \in \{1, 2\}, G_i(\sigma; \theta_i) < \max_{a^i \in N} G_i(a^i, \sigma^{-i}; \theta_i) - \epsilon | \pi \right\} \leq \delta, \quad (2.17)$$

where the probability is taken over the random joint type $\theta \sim \mu$, and the history of messages exchanged during the negotiation phase.

Let $c(\theta) \in \mathbb{R}^2$ be the “cooperative” payoff profile for joint type θ , that is, if the joint payoff profile is at least $c(\theta)$, then the players have successfully cooperated. (Multiple definitions of the cooperative profile can be considered.) For $\delta, \epsilon \geq 0$, π is (δ, ϵ) -compatible if

$$\Pr \{ \exists i \in \{1, 2\}, u(\theta_i, \sigma) < c_i(\theta) - \epsilon | \pi \} \leq \delta. \quad (2.18)$$

Finally, π is (δ, ϵ) -socially intelligent (SI) if it is both (δ, ϵ) -consistent and (δ, ϵ) -compatible.

Note that the *expected* payoff is bounded under the final mixed strategy profile σ , rather than the final joint action a itself. This is because consistency implies that σ must be a Nash equilibrium, and for an arbitrary matrix game there may be no pure-strategy equilibrium that helps to ensure maximal reward with high probability. The high-probability bound should be interpreted as ensuring that coordination succeeds for most of the type space, and is robust to “noise” in the negotiation process.

2.2.9 Feasible Cooperation

The fundamental idea behind this work is that if the human population achieves an average payoff of $M(\rho)$ in self-play, and $M(\rho)$ is (in some precise sense) socially optimal, then by observing two humans interact an AI should be able to infer enough about their private types to cooperate with them (though not necessarily achieving a payoff of $M(\rho)$) in the future. This can be leveraged by learning AI strategies that imitate the human for some number of steps $T \ll H$, and then use the information gained from the first T steps (extracted from the history h_T) to commit to a strategy $\pi(h_T)$ for which the human’s best-response (given their private type) will yield a “cooperative” joint strategy.

A critical point is that in general the AI cannot imitate a human perfectly forever, and eventually the human partner will realize that they are interacting with an AI. At this point the human’s behavior can become adversarial, but is still constrained to be rational. In this setting it is implied that even when the human is aware that they are interacting with an AI, their strategy must remain *Hannan consistent*. The AI can leverage this by committing to a strategy that prevents a rational human partner from “sabotaging” cooperation. Therefore cooperation can be defined in terms of the payoff profiles that are unilaterally *enforceable* by one player or the other.

Given a joint type $\theta = (\theta_1, \theta_2)$, and a (pure) strategy a_i , the ϵ -best response set for player $-i$ can be defined as

$$BR_{-i}^\epsilon(a_i; \theta) = \{a_{-i} \in A : G_{-i}(a_i, a_{-i}; \theta) \geq G_{-i}(a_i, \bar{a}_{-i}; \theta), \forall \bar{a}_{-i} \in A\} \quad (2.19)$$

Approximate best-responses are considered because over a finite time horizon a consis-

tent agent need never converge to a true best-response to its partner's strategy. A payoff profile $p = (p_1, p_2)$ is *feasible* given θ if there exists player $i \in \{1, 2\}$, and action $a_i \in A$ such that $p_i \leq \min_{a_{-i} \in BR_{-i}^\epsilon(a_i; \theta)} G_i(a_i, a_{-i}; \theta)$ and $p_{-i} \leq \min_{a_{-i} \in BR_{-i}^\epsilon(a_i; \theta)} G_{-i}(a_i, a_{-i}; \theta)$. This means that one player can choose a strategy such that, no matter how the other player chooses their ϵ -best response, both players receive payoffs at least as large as their payoff under p . Let $\mathcal{F}(\theta)$ be the set of all feasible payoff profiles under joint type θ , and define $\mathcal{P}(\theta) \subseteq \mathcal{F}(\theta)$ to be the subset of the feasible profiles that are not *strictly* Pareto dominated by some other feasible profile. The *cooperation profile* $c(\theta)$ has been defined such that $c_i(\theta) = \inf_{p \in \mathcal{P}(\theta)} p_i$, that is, the profile of minimal payoffs each player can expect to receive assuming a Pareto-efficient feasible profile is played.

2.2.10 Type Inference

We'll start with a simpler cooperation problem. Recall that ρ can be equivalently thought of as a distribution over human strategies, and a single behavioral strategy that is indistinguishable from this population. In the later interpretation, it can be said that ρ is (T, ϵ, δ) -socially intelligent using the definition from Section 2.2.3. Assume that ρ is indeed socially intelligent, and imagine that the AI first observes two humans following ρ under joint types sampled from μ . After observing the human's interact for T steps (yielding history T), the AI observes player 1's type θ_1 , and takes control of player 1's actions for the remaining H steps. The AI must commit to playing a pure strategy $\pi(h_T, \theta_1)$ for these H steps, while player 2 follows an arbitrary (H, ϵ', δ') -consistent strategy for the remaining H steps.

The question is, is there a mapping π such that with probability $1 - \delta_0$ over joint types and

observed histories h_T , the AI’s payoff will be $\geq c_1(\theta)$? In other words, can the AI extract enough information from the human (player 2’s) behavior to choose a strategy a_1 that will force them to cooperate?

It can be shown that $c_1(\theta)$ is itself too hard a target. The reason is that while $c(\theta)$ considers the entire game, which may include opportunities for “enforced” cooperation that the humans simply ignore. Instead, the best case is to achieve a payoff corresponding to cooperation in the “hardest” sub-game the humans could choose to play.

2.2.11 Conclusion & Major Takeaways of Social Cooperation

The problem of minimizing the population regret (Equation 2.11) is exactly as hard as Reinforcement Learning or Imitation Learning in the POMDP corresponding to ρ .

An alternative goal would be to minimize the AI’s regret relative to its worst-case performance given access to the human partner’s private type (θ_2). If the AI knows the joint type, then the fact that the human strategies are *consistent* with high probability means that the AI can simply choose a strategy that maximizes its own return under the human’s best response. *Stackelberg regret* can be defined as

$$R_M^{Stackelberg}(\pi, \rho) = \mathbb{E} \left[\max_{\sigma \in \Delta(A)} G_1(\sigma, BR_2(\sigma, \theta), \theta) - \frac{1}{H} M(\pi, \rho) \right] \quad (2.20)$$

where $BR_2(\sigma, \theta)$ is player 2’s best-response to σ under joint type θ , with ties broken so as to minimize player 1’s payoffs. A rough scheme for minimizing $R_M^{Stackelberg}(\pi, \rho)$ might work as follows:

1. Interact with ρ for d episodes to learn a model of ρ that is “sufficiently accurate” for the

first $T \ll H$ steps.

2. Against a new partner, imitate a strategy samples from ρ for T steps, and use the history h_T to estimate the partner's private type θ_2 .
3. Compute a mixed strategy σ maximizing player 1's payoffs under player 2's best response for their type θ_2 , and commit to σ for all future stages.

Because the human's strategy is consistent, they will eventually converge to their best response to σ .

What makes this useful is that imitating the human's policy over T steps may require far less data than imitating them over H steps. After T steps, even if the human is able to reliably distinguish the AI from another human, the AI will (hopefully) have enough information about θ_2 to manipulate the human into cooperating. It can be shown, however, that as with the population regret, there are cases where no strategy can achieve non-positive Stackelberg regret.

The key idea in this lower bound is that humans may be able to cooperate optimally without revealing their full private types. Therefore, an achievable notion of regret should account for exactly how much type information the human's strategy provides over the first T steps. This idea can be captured by restricting the space of AI strategies to the set Π_T that play an arbitrary strategy for the first T steps, and must then commit to a (possibly mixed) strategy σ for the remaining $H - T$ steps. Let the *manipulator regret* for $T < H$ be defined as

$$R_M^T(\pi, \rho) = \max_{\pi' \in \Pi_T} M(\pi', \rho) - M(\pi, \rho) \quad (2.21)$$

For $T \ll H$, if there is a model $\hat{\rho}_T$ of ρ that is accurate up to T steps, then ρ could be

approximated by a strategy that follows $\hat{\rho}_T$ for T steps, and then follows a fictitious-play strategy for the remaining $H - T$ steps. Computing a strategy minimizing $R_M^T(\pi, \rho)$ would then simply be a matter of planning in a POMDP.

The question is, to what extent can a sufficiently accurate model of ρ be learned from either online interactions (Reinforcement Learning) or offline observations (Imitation Learning)? This is non-trivial because in either case the private types of the humans are not directly observable. In the full horizon case ($T = H$), a model could be learned that simply marginalized over their private type.

To address this, a mapping from the human's private type θ_2 to their behavior can be learned. This is impossible, however, because it cannot be known a priori how much of the private type θ_2 the human's behavior conditions on, or if it conditions on the private type at all. Indeed, a model can always be found for ρ that is completely independent of θ_2 given θ_2 (and so cannot be used for inference), but nonetheless explains the data generated under ρ equally well.

To overcome this issue, if ρ does not depend on θ_2 at all, then typically $\max_{\pi' \in \Pi_T} M(\pi', \rho)$ will take on a much smaller value than if something could be inferred about θ_2 from observations of the human's behavior under ρ .

Since ρ is self-compatible, the long-term behavior of two agents sampled from ρ (over H steps) can be useful to infer their private types.

2.3 Assisted Exploration of Socially Cooperative AI Agents for Alignment

Social cooperation is important for AI agents deployed in the real world who can assist humans in achieving their goals without knowing *a priori* what those goals are in their shared environment. Current research on goal inference for AI assistance in Human-AI interactions assumes that humans will have a complete knowledge and understanding of the task environment. This may not be the case in many real-world applications such as driver-assistance autonomous systems or surgical robots, and certainly is not true in applications where part of the AI's role is to help the human acquire information, such as recommendation systems. Therefore, I propose a theoretical framework for AI assistance in a stateful environment with assistive Markov Decision Processes (MDPs) considering the extent to which an AI agent can aide a human in learning how to achieve their goals within the shared environment. I am proposing an *assisted exploration* paradigm where an AI agent can override a human's actions to support more efficient exploration of their environment without knowledge of the human's underlying goals. Within this paradigm, I have provided algorithms for selecting when and how to intervene in MDPs with finite state and action spaces. I have investigated into polynomial-time regret bounds for consistent assisted exploration that hold for a large class of possible human learning strategies and allow sufficient exploration by AI. It helps to make the AI Value Alignment task efficient by bounding the number of unnecessary interventions made by the AI, to ensure that disruption of the human's intended behavior is kept to a minimum.

2.3.1 Introduction

The Assistive Multi-Armed Bandit [59] presents the *assistive* bandit framework, in which a human recommends arms to pull in a multi-armed bandit, but the AI decides which arms actually get pulled. The challenge is that the AI never observes the payoffs received for each arm, only the human does. They prove that it is possible to design AI policies such that the human-AI team is *Hannan consistent* in two general cases:

1. The human knows the payoff distributions, and selects the best arm most frequently.
2. The human is (noisily) greedy, choosing the arm with the highest empirical payoff most often.

They also provide an information theoretic lower bound on the mutual information between the human’s recommendations and the index of the optimal arm, in the Bayesian case of the Beta-Bernoulli MAB. They present the results of some computational experiments supporting their theoretical results, but no human-subjects experiments appear.

2.3.2 Assistive Markov Decision Processes

In this section I have extended the assistive bandit model [59] to the sequential setting, and show that many of the results for the bandit setting hold in modified form. I assume an underlying *episodic* MDP $\mathcal{M} = \{S, A, P, R, p_0, H\}$, where S and A are the state and action spaces, P is the state transition distribution, R is the reward function, p_0 is the initial state distribution, and H is the episode length. I also assume here that S and A are finite and that P , R and p_0 are initially unknown to either the human or the AI. At time step t , the human “recommends” an action h_t to

the AI, which then chooses an action a_t to take in the environment. The human and the AI then observe the next state s_{t+1} (or a new state sampled from p_0 if the episode has ended), while the human observes the AI's action a_t , and the reward $r_t = R(s_t, a_t)$. The AI does not observe this reward signal.

Consistency The first focus is to define what it means for the human-AI team to form a *consistent* learner. Let Q_i^* and V_i^* denote the optimal Q -function and value function for \mathcal{M} (these depend on the time-step i because the MDP has a fixed time limit). For episode k , let $t(k)$ denote the time step at which the episode started, such that $t(1) = 1$, and $t(k+1) - t(k) = H$. Let $\bar{R}_k = \sum_{t=1}^{t(k+1)-1} r_t$ denote the total reward received up to and including episode k , and let $R^* = \mathbb{E}_{p_0}[V_i^*(s)]$ denote the optimal expected total reward. It can then be said that the human-AI team is *consistent* if

$$\lim_{k \rightarrow \infty} \frac{1}{k} \mathbb{E}[\bar{R}_k] = R^* \quad (2.22)$$

which means that the expected average return per episode converges to the expected return of the optimal policy in the limit. Here the expectation is taken over all possible histories of interaction of length $t(k+1) - 1$.

Noisily Optimal Policies Assume that the human follows a fixed, stochastic policy π^H . I define the *set-valued* optimal policy as $\pi_i^*(s) = \arg \max_{a \in A} Q_i^*(s, a)$. π^H is *noisily optimal* if $\arg \max_{a \in A} \pi_i^H(a|s) \subseteq \pi_i^*(s)$ for all $s \in S$ and $i \in [1, H]$. This means that the actions most frequently chosen by the human in any state are optimal for that state, even though the human may choose sub-optimal actions with some non-zero probability. Note that the Boltzmann-rational policy where $\pi_i(a|s) \propto e^{Q_i^*(s,a)}$ would be noisily optimal. The next theorem then follows from

Proposition 1 of [59]:

Theorem 2.5. *Assume that the human’s policy π^H is noisily optimal for \mathcal{M} , then if the AI always takes the most frequently recommended arm in state s (with ties broken randomly), the human-AI team will be consistent for \mathcal{M} .*

Proof Sketch: The probability of receiving a sub-optimal payoff in episode k is equal to the probability of taking a sub-optimal action in any state during episode k . Treating each state as a separate assistive bandit, by Proposition 1 of [59], the average expected probability of this occurring at any given state s goes to zero asymptotically, and since S is finite so too does the average expected probability of making any mistake within an episode go to zero. QED.

2.3.3 Noisy Model-Based Learners

If human agents are noisily optimal, then they effectively know both the transition probabilities and the reward function of \mathcal{M} . The more interesting question is what happens when the human has to learn these online with the help of the AI. Rather than modeling the inner workings of the human’s learning process which can be intractable and computationally expensive, I compare their asymptotic behavior to that of a simplified learning process. Let \hat{P}_k and \hat{R}_k be tabular, empirical estimates of the transition probabilities and rewards based on the history up to, but not including episode k . For state-action pairs that have not been observed so far, these can be initialized pessimistically, such that any unobserved action yields an immediate reward of zero, and transitions to an absorbing state with reward zero as well. I define \hat{Q}_k and \hat{V}_k as the Q -function and value function corresponding to this empirical model.

Let $\bar{\pi}_k^H$ be the human’s policy for episode k . The human is *noisily greedy* if, for some k_0 ,

for all episodes $k > k_0$, $\arg \max_{a \in A} \bar{\pi}_k^H(a|s) \subseteq \arg \max_{a \in A} \hat{Q}_k(s, a)$. This means that *eventually* the human will have a strictly higher probability of pulling at least one optimal arm than they have of pulling any sub-optimal arm under the current empirical model.

Theorem 2.6. *If the human is noisily greedy, then there exists an AI strategy such that the human-AI team is consistent in \mathcal{M} .*

Proof Sketch: Analogous to the structure of Proposition 2 of [59], where the AI follows an independent, exploratory strategy with decreasing probability at each episode. It explores often enough that the empirical Q -function \hat{Q}_k eventually converges to the true Q -function, such that if the human is noisily greedy, then they are following a noisily optimal policy w.r.t. the true Q -function. The proof of Theorem 2.2 with Lemma 2.1 then shows that if, during non-exploration episodes, the AI selects the human's most frequently recommended action in each state, then the resulting sequence of policies will be consistent.

Q-Learning Unlike the bandit setting, implementing noisily greedy behavior is non-trivial in an MDP. For the human to do this, they may have to maintain a complete model of the MDP internally, and plan against this at every new episode. As this potentially represents a large cognitive load for the human, it is an open research problem if noisily greedy behavior can be approximated via a less complex algorithm.

Now let \hat{Q}_k represent a tabular estimate of the optimal Q -function updated using a Q -learning rule with a learning rate of α . Can it be shown that if the human bases their behavior on this *model free* estimate that it is still possible for the AI to achieve consistent behavior?

Theorem 2.7. *If, for all episodes $k > k_0$ the human's policy is noisily optimal w.r.t. the Q -learning estimate \hat{Q}_k , then there exists an AI strategy such that the human-AI team is consistent.*

2.3.4 On-Policy Learners

Both model-based learning and Q -learning are *off-policy* algorithms, which allow them to identify optimal behavior regardless of whether the trajectories they observe were generated under their own policy (so long as these trajectories sufficiently explore the state-action space). This allows the AI to modify the human's actions without worrying about how modifying the human's actions might affect their learning process. A more difficult case is where the human uses an *on-policy* algorithm (e.g., SARSA [255, 295]), that estimates the value of an immediate action based on the assumption that it will continue to follow the same policy for the rest of the episode, rather than what it believes the optimal return for a state will be. If the AI intervenes too frequently, the human's Q -function estimate, and the corresponding policy, may never converge to the optimum.

2.3.5 Assisted Exploration

In Section 2.3.2, I define the AI agent such that it is allowed to inject exploration episodes on a fixed schedule, ignoring the information collected under the human's own policy, which may itself involve some degree of exploration. In this section I derive an AI intervention strategy that only explores when necessary to ensure the human has a complete understanding of the task environment. As before, I assume that the underlying task is an episodic MDP $\mathcal{M} = \{S, A, P, R, s_0, H\}$ with a finite number of states and actions. To simplify later notation, without loss of generality I replace the initial state distribution p_0 with a unique starting state $s_0 \in S$. I assume here that the human has complete knowledge of the reward function R , but does not initially know the transition probabilities P . Another assumption is that the AI has no

knowledge of R , and can never directly observe the rewards. Two cases considered include the case where the AI knows P exactly, and when it does not know P initially.

In this section I also isolate the issue of noise in the human’s behavior from that of insufficient exploration, by assuming that the human’s behavior will be asymptotically optimal relative to an empirical model \hat{P}_k estimated from the history h_k of state-action trajectories up to and including episode k . Driven by an interest in finite time regret guarantees, some assumptions have to be made about how quickly the human’s behavior becomes optimal (relative to the model \hat{P}_k). For a policy π , define V^π as the state-value function under π and the true MDP \mathcal{M} , and define \hat{V}_k^π as the state-value function under π and the MDP $\hat{\mathcal{M}}$ obtained by substituting the model \hat{P} for the true transition probabilities P . Let π_k be the policy followed by the human in episode k .

Definition 2.7. The human is *polynomially empirically optimal* (PEO) if there exists a function f such that $f(k)$ is $O(\text{poly}(S, A, H, k))$, and $\mathbb{E}[\hat{V}_{k-1}^*(s_0) - \hat{V}_{k-1}^{\pi_k}(s_0)]$ has an upper bound.

Note that the fact that empirical optimality says nothing about whether the human’s strategy would achieve small regret on the underlying task, as the empirical model \hat{P} could be incomplete. It can be shown however, that with the right choice of intervention strategy, the AI can ensure that the empirical model is close to the true MDP, and that such a strategy can be defined without knowledge of the human’s reward function R .

Theorem 2.8. *Assume that the human’s strategy is PEO, then there exists an AI intervention strategy such that the human-AI team will be polynomially consistent.*

Proof sketch: Using the well-known high-probability bound on the L_1 -error of empirical probability intervals [327], a confidence interval can be defined around $\hat{P}(\cdot|s, a)$ for each s and a , and from this, a confidence bound can be derived on \hat{V} . So long as $|\hat{V}(s_0) - V(s_0)|$ goes to zero

sufficiently fast, then the regret of the human-AI strategy will go to zero as well. This is helpful to derive the intervention for the AI agent to assist humans.

Conclusion

In this section I present some simple experimental scenarios highlight the advantages of this more general formulation, and provide some straightforward theoretical results stating that most relevant formulations of reward learning can be reduced to assistance problems (i.e., dec-POMDPs). Some advantages of Assistive AI Agents Over Reward Learning [270] for AI alignment is based on the theory that definitions of “reward learning” problems (such as IRL) are best modelled as *assistance* problems, that is, as decentralized partially observable Markov Decision Processes (dec-POMDPs) where the parameters of the reward function are only known to the human player.

Chapter 3: Learning to Cooperate in Populations of Socially Rational Agents

AI agents deployed in the real world must be able to reliably *cooperate* with humans (as well as other, heterogeneous AI agents). To provide formal guarantees of successful cooperation, some assumptions about how these partner agents could plausibly behave are required. Realistic assumptions must account for the fact that other agents may be just as adaptable as the AI agent is. In this Chapter, consider the setting where an AI agent must cooperate with members of some *target population* of agents in a finitely repeated two-player general-sum game, where individual utilities are private. Two natural assumptions in this setting are 1) all agents in the target population are individually rational learners, and 2) when paired with another member of the population, with high-probability the agents will achieve the same expected utility as they would under some Pareto-efficient equilibrium strategy of the underlying stage game. The theoretical results show that these assumptions alone are insufficient to select an AI strategy that achieves *zero-shot* cooperation with members of the target population. Therefore, consider the problem of learning such a cooperation strategy using observations of members of the target population interacting with one another, and providing upper bounds on the sample complexity of learning such a cooperation strategy. The main result shows that, under the above assumptions, these bounds can be much stronger than those arising from a “naive” reduction of the problem to one of imitation learning [18].

3.1 Introduction

Imagine a hospital deploying an AI assistant to help their surgeons plan and execute complex surgeries. For instance, the AI assistant can take the role of a second surgeon in complex procedures that benefit from a two-surgeon approach [168]. When first deployed, the AI agent is unlikely to have comprehensive knowledge of the population consisting of its potential partners (i.e. the surgeons working in the hospital). Differences between human surgeons, such as preferences, capabilities, and internal states—including surgical experience, familiarity with specific procedures, or even mental focus under pressure—can critically impact cooperation. A successful AI agent should be able to adapt its strategy to each human surgeon it partners with. The central question of this Chapter in my Thesis is how to efficiently learn such adaptive and cooperative *meta-strategies* from a dataset of cooperative interactions between the members of the target population.

To illustrate, consider experience level. An experienced surgeon may prefer a fast strategy to reduce surgery duration, improving post-op recovery time. Conversely, an inexperienced surgeon may prefer a slower, cautious strategy. Here a *strategy* refers to the policy an agent follows in a single collaborative surgery. My goal has been to learn an adaptive *meta-strategy* that maps from the history of interactions (e.g. history of collaborative surgeries performed so far) to strategies, which allows for the AI agent to adapt to the needs of its current human partner over time.

It is possible to learn a good AI *strategy* for individual partners using past surgical data through imitation learning (e.g. [173]). However, learning a good *meta-strategy* through imita-

tion becomes impractical as task complexity, partner diversity, and task duration increase. Imitation learning here would mean learning a function mapping from histories of multiple surgeries to new surgical strategies. To do so, the AI agent would require datasets that capture long-term interactions between human surgeons and cover the full range of surgeon and patient profiles. Additionally, in high-stakes environments like surgery, imperfect imitation may lead to unacceptable failure modes, resulting in the AI agent's role being terminated.

The problem setup. To formalize the above intuitions, model the interaction between the AI agent and the individual members of the population as a repeated, two-player, general-sum matrix game with private types. Each agent's type is their private information, where different types of agents have distinct payoff functions. Types embed behavioral differences amongst the agents through payoffs, inducing general-sum games between partners with different types, even though they are collaborating on a task that requires teamwork (e.g., no-conflict games [9] or ad-hoc teamwork [283]). Each stage game represents a complete interaction between two agents. For instance, in the case of the surgery example, a single stage game of the repeated game corresponds to a complete surgery.

Contributions. A novel definition called a *socially intelligent* population is presented, where the member agents are (1) Consistent, meaning an individual will perform at least as good as its best pure strategy in hindsight and (2) Compatible in pairs, meaning they achieve performance comparable to a Pareto-optimal Nash equilibrium (PONE). The former, also called the no-regret property, is often seen as a minimal requirement for rationality, whereas the latter has been used to describe successful cooperation by [239]. The primary contribution is an upper bound on the sam-

ple complexity for learning cooperation meta-strategies in socially intelligent populations. While consistent partners (as defined in Section 3.3) do not guarantee success for imitation learning, it is demonstrated that compatibility within the population makes imitation feasible. However, the lower bound on sample complexity grows exponentially due to the need to account for histories. An upper bound in Theorem 3.6 shows that for socially intelligent partners—who are both consistent and compatible—it is possible to learn cooperation meta-strategies more efficiently than through imitation alone. A consequence of the lower bound in Theorem 3.4 is that, even when the target population can be assumed to be socially intelligent, without additional information about this population a strategy that can reliably cooperate with members of this population cannot be found. Therefore, consider a more realistic interpretation of the zero-shot coordination problem, where the AI agent must cooperate with an entirely new partner (whose private type is unknown), but has observations of other members of the population, so it can learn the strategies (perhaps better thought of as "meta-strategies") that these agents use to coordinate with new partners.

Outline. In Section 3.2, the intuition of this approach and the motivations behind it are discussed. Then in Section 3.3, the Multi-Agent setting is defined with a background on consistency (in the form of Hannan-consistency) and external regret. A novel definition of compatibility is given in definition 2.2, inspired by [239]. Section 2.2.3 introduces the proposed definition of social intelligence and presents a realistic class of agents that meet this criterion. In Section 3.4, this learning problem is framed as minimizing *altruistic regret* and derive lower bounds on its sample complexity. Finally, in Section 3.6, the upper-bound result is presented, proving that a *imitate-then-commit* strategy can leverage the social intelligence of the population to learn cooperative strategies more efficiently.

3.2 Motivation

Socially intelligent populations. My focus in this Thesis is on populations of agents where members have established conventions that enable effective cooperation. For instance, two surgeons can plan and execute a complex surgery together efficiently, even if they have not worked together before, because they share a common set of conventions learned through similar education. This can be seen as the members of the population being *compatible* with each other. In addition, each member should individually satisfy a base level of rationality. My definition of social intelligence formalizes these intuitions.

General-sum games. In this setting, agents with different types will have distinct payoff functions due to different behavioral propensities. Consider the example given in Table 3.1 for a pair of experienced and inexperienced surgeons. Even though the surgery is a cooperative task, the agents have non-identical payoffs due to differences between their types (i.e. experience level). Here, the general-sum aspect models the potential failure of coordination between the agents due to their private types. If the row player is experienced and mistakenly thinks its partner is also experienced, it will choose the fast approach, leading to the sub-optimal cooperation outcomes. However, if, for instance, the agents learned each other's types through repeated interactions, they can both choose the cautious approach towards the optimal cooperation outcome (Cautious, Cautious).

This approach proposes that the AI agent initially mimics the behavior of a team member over a short horizon, gathering enough information to infer its partners' types, while behaving as expected from a team member. For instance, the AI agent can start by imitating the average

	Fast	Balanced	Cautious
Fast	(4, 2)	(5, 4)	(3, 3)
Balanced	(5, 4)	(6, 6)	(4, 5)
Cautious	(3, 3)	(4, 5)	(7, 7)

Table 3.1: Payoff matrix for a repeated two-player game with an experienced surgeon (row) and inexperienced surgeon (column). The best cooperative outcome is achieved when experienced surgeon slows down to match the inexperienced.

behavior of a human surgeon from the dataset, gradually inferring the human partner’s type. This preliminary imitation might not be immediately efficient for the specific partner, but as long as it remains human-like, it is more likely to be tolerated. Once the partner’s type is inferred, the AI agent can transition to a type-conditioned strategy that is well-aligned with its partner. This approach would ensure that the partner is more likely to engage with the AI agent as a trusted collaborator, avoiding early-stage friction that might otherwise lead to the termination of the AI agent’s involvement. Formalization of the repeated two-player general sum matrix games setting is motivated from the notion of replicator dynamics [37, 266] in evolutionary game theory. The replicator equation represents the proportion of each type in a population as the difference of the fitness of a population for that type to the average fitness across all types. The replicator dynamics construct helps to understand the type of the two agents sampled for finitely repeated two-player general sum matrix games.

These theoretical results apply to various real-world scenarios where the goal of an AI agent is to learn how to cooperate with self-interested agents with private types such as humans. Most importantly, the AI agent itself is not necessarily self-interested, since its goal is to assist or cooperate with partners coming from a population. However, the partners it is trying to cooperate with are self-interested. This framework offers efficient bounds for learning viable cooperation meta-strategies based solely on observed interactions between the members of the population.

Further insightful examples for interacting with populations are given in Section 3.7 e.g. AI Health Coach and human-robot cooperation in a factory.

3.3 Preliminaries

As in Section 2.1.6, I model one-on-one interactions between two members of a socially intelligent population as a repeated two-player matrix game with private types.

3.3.1 Socially Intelligent Agents

As in Chapter 2, I require that members of a socially intelligent population be both consistent and compatible. I refer to Sections 2.2.4 and 2.2.5 for the formalization of consistency and compatibility, respectively. This forms the basis of Socially Intelligent populations of Agents, described in Section 2.2.6.

3.4 Learning to Cooperate

It can be assumed that the agent (henceforth referred to as the “AI agent”) will take the role of agent 1, while the other agent (referred to as the “partner”) will be agent 2. The objective is to choose a meta-strategy for the AI agent that can cooperate with a partner drawn from some *target population* nearly as effectively as agents from this population cooperate with one another. For the class of games $\mathcal{G} = (\mathcal{I}, \mathcal{A}, \Theta, G, T)$ as defined in Section 3.3, the target population can be defined as a set \mathcal{C} of agents forming a (δ, ϵ, T) -SI class with respect to Θ . Ideally, the goal is to choose an AI meta-strategy π which can cooperate with \mathcal{C} *without* any additional information

about the strategies in \mathcal{C} . Looking at the handshake protocol example in Section 2.2.6, in many cases a population is likely to use arbitrary conventions to coordinate their behavior, and intuitively cooperation is impossible without prior knowledge of these conventions. (This intuition is formalized in Theorem 3.4).

Therefore, the problem of learning a cooperative meta-strategy is outlined with prior observations of members of the target population interacting with one another. A *social learning problem* is defined by a tuple $\{\mathcal{G}, \mathcal{C}, \rho, \mu\}$, where \mathcal{C} is the target population (SI w.r.t. Θ), ρ is a distribution over \mathcal{C} , while μ is a distribution over the joint type space $\Theta \times \Theta$. \mathcal{C} can be the set of possible strategies that any member of the target population might follow, while ρ is the frequency of those strategies within the population. To choose an AI strategy, a dataset $\mathcal{D} = \{(\theta_1^j, \theta_2^j, h_T^j) | j \in [n]\}$ can be leveraged covering n episodes of length T . In each episode j , two agents π_j^1 and π_j^2 are sampled independently from ρ , and played together under the joint type $\theta^j \sim \mu$. The AI agent observes the full history h_T^j , along with the agents' types θ_1^j and θ_2^j . A specific learning algorithm is denoted as a data conditioned strategy $\pi(\mathcal{D})$.

3.4.1 Altruistic Regret

An AI strategy has been considered that minimizes the regret relative to some Pareto optimal solution to $G(\theta)$. Rather than minimizing regret in terms of the AI's own payoffs, however, the *partner's* regret can be minimized relative to their (worst case) PONE in $G(\theta)$. This regret can be formalized as follows:

Definition 3.1 (Altruistic Regret [18]). Let $(\sigma_i^*, \sigma_{-i}^*)$ denote the PONE with the *lowest payoff* for

the agent $-i$ where $i \in \{1, 2\}$. The altruistic regret of agent i is defined as

$$R_i^{\text{alt}}(h_T; \theta_{-i}) = \sum_{t=1}^T G(\sigma_i^*, \sigma_{-i}^*; \theta_{-i}) - G(a^i(h_t), a^{-i}(h_t); \theta_{-i}). \quad (3.1)$$

In practical cooperation tasks, outcomes that have low regret for the partner will have low regret for the AI agent as well.

The cooperation objective for the AI agent can then be formalized as minimizing the altruistic regret. Unlike the definition suggests, the AI agent must know its own type as well. This is due to the fact that as seen in the handshake protocols example, if the AI agent fails to imitate a human of its type or fail to communicate its type correctly, the partner might switch to a safe strategy.

The goal for the AI agent is to minimize its *expected* altruistic regret over partners sampled from ρ and types sampled from μ . The following lemma shows that the problem of minimizing regret with respect to a heterogeneous population \mathcal{C} can be formulated as that of minimizing regret w.r.t. a single stochastic strategy.

Lemma 3.1. *Let \mathcal{C} be a finite set of agents that are (δ, ϵ, T) -socially intelligent w.r.t. type space Θ , and let ρ be a distribution over \mathcal{C} . There exists a mixed strategy $\bar{\rho}$ that forms an (δ, ϵ, T) -socially intelligent class, and which is equivalent to playing against partners sampled from ρ in expectation.*

Proof. In a perfect recall game, every behavioral strategy has an equivalent mixed strategy and vice-versa [11]. Thus ρ can equivalently be defined as a distribution over mixed strategies so that $\rho \in \Delta(\Delta(N))$. Then defining $\bar{\rho}(a) = \int_{\Delta(N)} \sigma(a) d\rho(\sigma)$ where $a \in [N]$ denotes a pure strategy (i.e. action) completes the proof.

In order to show the joint impact of consistency and compatibility on the learning problem, cases where the population is either consistent or compatible, but not both, are highlighted.

3.4.2 Consistency without Compatibility

Assume that \mathcal{C} consists of agents that are consistent but not necessarily compatible. The most general class in this case is the class of all no-external-regret learners (no-regret henceforth). It is a well-established result that the long-run average of no-regret learning converges to the set of coarse correlated equilibria. The question is whether the AI agent can learn to do better than a coarse correlated equilibrium when paired with a member of \mathcal{C} , using only a dataset \mathcal{D} that consists of histories of play for different Coarse Correlated Equilibria (CCE).

Theorem 3.2. *There exists a consistent yet incompatible class of agents \mathcal{C} such that even with an infinite amount of data, in the worst-case, the AI agent suffers constant altruistic regret.*

Proof. The proof follows from the theorem 5.1 of [217] which shows that given any coarse correlated equilibrium of a two-player normal-form game, there exists a pair of no-regret learners that would converge to it. Since \mathcal{C} can be any subset of no-regret learners, those who converge to inefficient CCE cannot be excluded. If the class \mathcal{C} contains only the agents that converge to Pareto-inefficient CCE, it will not be possible to learn optimal strategies from any dataset. For example, consider the payoff matrices given in table 3.2 for two pairs of types. Here, the payoff of each agent depends only on its partner. In both games, there exists only one PONE with payoffs (3, 4). However, every pure and mixed strategy profile is a CCE in both matrices. In the worst case, there may be a class of agents \mathcal{C} which only converge to the CCE (B, B) . More importantly, the behavior of each agent does not need to carry any information about their type, since each

	<i>A</i>	<i>B</i>
<i>A</i>	3, 2	1, 2
<i>B</i>	3, 4	1, 4

(a) The game matrix for types (θ_1, θ_2)

	<i>A</i>	<i>B</i>
<i>A</i>	3, 4	1, 4
<i>B</i>	3, 2	1, 2

(b) The game matrix for types (θ_3, θ_4)

Table 3.2: A class of games where an agent’s payoff depends only on its partner.

agent’s payoff depends solely on its partner. When the AI’s strategy is deployed, it will face a partner drawn from \mathcal{C} whose type is unknown, regardless of the imitation demonstrations dataset. Since type cannot be inferred from behavior any more, there is no way for the AI agent to know which of the two game matrices it is playing. At best, the AI (row player) can choose one of the two PONE with uniform probability and commit to it. There is a 0.5 probability that the AI will play the wrong PONE, incurring constant altruistic regret. \square

3.4.3 Compatibility without Consistency

Let it be assumed that the members of \mathcal{C} are compatible, but not consistent. In Section 2.2.6, such a class can be constructed with agents using a handshake protocol to exchange type information, and then playing the agreed-upon PONE of the current game. However, if at any any time an agent deviates from this chosen solution, there is no restriction on what strategy each agent will follow from that point forward. The members of \mathcal{C} may even employ grim-trigger strategies that “punish” any mistake on the part of the other agent by following a highly sub-optimal strategy. Even if at some point in the future they could potentially switch back to a cooperative strategy (i.e., forgive the other agent), this may not occur within the finite horizon T . A single mistake at any time on the part of the AI agent may yield the maximal altruistic regret for the remainder of the interaction. The AI must therefore learn to imitate at least one member of \mathcal{C} perfectly using the dataset \mathcal{D} , and the problem of learning to cooperate reduces to imitation learning (specifically

the *no-interaction* setting of [245]).

A lower bound on the altruistic regret can be derived in this case by considering a game in which there is only a single type (such that individual payoffs are common knowledge), and each agent’s payoffs depend only on their own actions. Specifically, the first $N - 1$ actions each yield a payoff of 1, regardless of the other agent’s action, while the N th action yields a payoff of 0. In this game, a compatible class \mathcal{C} can be constructed such that, for the first $k \leq T$ steps, the agents execute some “authentication protocol”, which allows them to identify other agents following strategies in \mathcal{C} . For the first $k - 1$ steps, each agent samples one of the first $N - 1$ actions, with the sequence of actions forming a *challenge code* that the other agent must respond to by selecting the correct action at step k . If an agent’s partner fails to provide the correct response at step k , the agent will follow the N th action for the remaining $T - k$ steps, such that it receives no further payoff from that point forward. Using such strategies, and an approach similar to that of [245], a lower bound on the altruistic regret can be derived as a function of the number of samples in the dataset $|\mathcal{D}|$.

Theorem 3.3. [18] *Let $K = |\mathcal{D}|$ be the number of interaction histories in the dataset. For any $k < T$, and any $\delta, \epsilon \geq 0$, there exists a class of games \mathcal{G} , and class \mathcal{C} of (δ, ϵ, T) -compatible agents such that, for any data-dependent meta-strategy $\hat{\pi}(\mathcal{D})$, the altruistic regret is lower-bounded as*

$$E [R_i^{alt}(h_T; \theta_{-i})] \geq \frac{T - k}{e} \frac{N - 2}{N - 1} \min \left\{ \frac{1}{2}, \frac{(N - 1)^{k-1} - 1}{2K + 1} \right\}, \quad (3.2)$$

where the expectation is taken over h_T , θ , and \mathcal{D} . Then, for small altruistic regret, the sample complexity grows exponentially in k .

Proof sketch. Let an “authentication” function $f : [N - 1]^{k-1} \mapsto [N - 1]$ be chosen

that maps each possible $(k - 1)$ -step history of actions to a specific action in $[N - 1]$. Then, a class \mathcal{C} can be constructed, consisting of a single meta-strategy that, for the first $k - 1$ steps selects its actions so that the initial $k - 1$ step history of its actions is distributed according to a specific, nearly uniform distribution μ . At step k , agent i chooses action $f(h_{k-1}^{-i})$, where h_{k-1}^{-i} is the sequence of actions chosen by the other agent $-i$. So long as agent $-i$ response with the correct action $f(h_{k-1}^i)$ at step k , agent i will continue to choose actions in $[N - 1]$. Therefore, in self-play \mathcal{C} will be (δ, ϵ, T) -compatible for any $\delta, \epsilon > 0$.

The AI agent, however, is unaware of f , and must estimate this function from \mathcal{D} . If the AI's strategy fails to correctly authenticate at step k , its partner will switch to the N th action, which yields a payoff of zero, such that the AI will suffer an altruistic regret of $T - k$. Because f is deterministic, a meta-strategy found via imitation learning will correctly authenticate for any history h_{k-1}^{-i} found in \mathcal{D} , but has a probability of $1 - 1/(N - 1)$ of failing to authenticate for an unseen history. By sampling from a carefully chosen distribution μ , the probability of encountering an unseen history is greater than $\min\{\frac{1}{2}, [(N - 1)^{k-1} - 1]/(2K + 1)\}$, which leads immediately to the lower-bound on the expected altruistic regret. Note that any $k < T$ can be chosen so as to maximize this lower bound for any values of T, N and K .

3.5 Lower Bound for Socially Intelligent Populations

Theorem 3.4. *Let $K = |\mathcal{D}|$. For any $\delta, \epsilon > 0$, there exists a class of games \mathcal{G} , and class \mathcal{C} of (δ, ϵ, T) -socially intelligent agents such that, for any data-dependent meta-strategy $\hat{\pi}(\mathcal{D})$, the*

altruistic regret is lower-bounded as

$$E [R_i^{alt}(h_T; \theta_{-i})] \geq \Omega \left((T - k) \min \left\{ \frac{1}{2}, \frac{(N)^{k-2} - 1}{2K + 1} \right\} \right), \quad (3.3)$$

for some $k \geq T\epsilon$.

Proof sketch. Similar to the proof for Theorem 3.2, a class of games can be defined in which players must exchange their private types to be compatible, while at the same time they can implement consistent behavior without revealing anything about their types. A socially intelligent class of agents can be constructed for this class of games. These agents can be augmented such that they implement a k -step authentication protocol (as in Theorem 3.3) before switching to the socially intelligent meta-strategy if authentication succeeds. If authentication fails, the agents will switch to some alternative consistent meta-strategy. So long as $k - 1 \leq T\epsilon$, the resulting class of agents will be (δ, ϵ, T) -socially intelligent. As discussed in Section 3.4.2, a consistent meta-strategy may never communicate an agent's type. Without knowing its partner's type, the AI agent may suffer arbitrarily large altruistic regret at each step, as it cannot identify the actions that will maximize its partner's utility.

3.6 Upper Bound for Socially Intelligent Populations

A key idea behind this work is that against a socially intelligent target population, rather than trying to perfectly imitate a member of the population throughout the entire episode, the AI agent only needs to imitate them long enough to learn about its partner's private type. Once it has this information, the AI agent can leverage the fact that the partner's strategy is consistent

Algorithm 1 The \tilde{T} -step *imitate-then-commit* meta-strategy (denoted by $\pi_{\tilde{T}}^{IC}$). It is assumed here that the AI acts as agent 1.

- 1: **Inputs:** Interaction dataset \mathcal{D} , imitation time \tilde{T} .
 - 2: Initialize the imitation policy $\hat{\pi}_{\tilde{T}}^1(D)$.
 - 3: **for** step $t = 1, \dots, \tilde{T}$ **do**
 - 4: Execute action $a_t^i \sim \hat{\pi}_{\tilde{T}}^1(h_t; D)$
 - 5: **end for**
 - 6: **for** action $j \in N$ **do**
 - 7: $z_j = \sum_{i \in N} \hat{z}(h_{\tilde{T}})_{i,j}$
 - 8: **for** action $i \in N$ **do**
 - 9: $x_j(i) = \hat{z}(h_{\tilde{T}})_{i,j} / z_j$
 - 10: **end for**
 - 11: **end for**
 - 12: Sample $x = x_j$ with probability z_j
 - 13: **for** step $t = \tilde{T} + 1, \dots, T$ **do**
 - 14: Execute action $a_t^i \sim x$
 - 15: **end for**
-

against *any* strategy, and try to “coerce” the human partner into playing a strategy that minimizes the altruistic regret. Such meta-strategies are henceforth referred as *imitate-then-commit* (IC) strategies, which use the previous observations \mathcal{D} to learn an imitation strategy that it follows for the first $\tilde{T} < T$ steps of the interaction. In this Section, an upper bound is provided on the expected altruistic regret of a specific (IC) meta-strategy, as a function of the number of episodes in \mathcal{D} , subject to the following assumptions:

Assumption 3.1. [18] For $\delta_0, \delta_1, \epsilon_0, \epsilon_1 > 0$, and $\tilde{T} < T$, it can be assumed that

1. ρ is $(\delta_0, \epsilon_0, T)$ -consistent.
2. ρ is $(\delta_1, \epsilon_1, \tilde{T})$ -compatible.

Imitation learning. Under an imitate-then-commit meta-strategy, the sample complexity is defined entirely by the number of episodes the AI agent needs to observe to learn a good \tilde{T} -step imitation policy. Fortunately, imitation learning is a well-studied problem, whose existing com-

plexity bounds can be leveraged. The one caveat is that this setting requires bounds on the total variation distance between the distribution over the partial history $h_{\tilde{T}}$ under the population strategy ρ , and under the learned strategy. Given the dataset \mathcal{D} , the imitation strategy can be defined $\hat{\pi}_{\tilde{T}}^1(\mathcal{D})$ such that $\hat{\pi}_{\tilde{T}}^1(h; \mathcal{D})$ is the empirical distribution over agent 1's actions for each history h occurring in \mathcal{D} , while $\hat{\pi}_{\tilde{T}}^1(h; \mathcal{D})$ is the uniform distribution over N for $h \notin \mathcal{D}$. The *marginal* imitation strategy can be defined as $\hat{\pi}_{\tilde{T}}^1 = \mathbb{E}_{\mathcal{D}}[\hat{\pi}_{\tilde{T}}^1(h; \mathcal{D})]$, where the expectation is taken over the sampling of the dataset \mathcal{D} itself. Then, then the following bound on the distribution of $h_{\tilde{T}}$ under the imitation strategy can be proven:

Lemma 3.5. *Let $p_{\tilde{T}}$ be the distribution over partial histories $h_{\tilde{T}}$ under the population strategy ρ paired with itself, and let $\hat{p}_{\tilde{T}}$ be their distribution under $\hat{\pi}_{\tilde{T}}^1$ paired with ρ . Then,*

$$\|p_{\tilde{T}} - \hat{p}_{\tilde{T}}\|_{TV} \leq \min \left\{ 1, \frac{N^{2(\tilde{T}+1)}\tilde{T} \log(K)}{K} \right\}, \quad (3.4)$$

where $K = |\mathcal{D}|$.

Proof of Lemma 3.5 First Theorem 4.2 of [245] is applied, which states that, for episodic imitation learning over H -step trajectories, for any expert policy π^* , there is

$$J(\pi^*) - \mathbb{E}_{\mathcal{D}} [J(\hat{\pi}_{\tilde{T}}^1(h; \theta, \mathcal{D}))] \leq \min \left\{ H, \frac{|S|H^2 \log(K)}{K} \right\}, \quad (3.5)$$

where S is the state space, with per-step rewards bounded in $[0, 1]$. This interaction can be modeled with ρ as a \tilde{T} -step episodic MDP/R with $S = \mathcal{H}_{\leq \tilde{T}}$. Plugging in $H = \tilde{T}$, $|S| < N^{2(\tilde{T}+1)}$,

and $\pi^* = \rho$ gives us

$$J(\rho) - \mathbf{E}_{\mathcal{D}} [J(\hat{\pi}_{\tilde{T}}^1(h; \theta, \mathcal{D}))] \leq \min \left\{ \tilde{T}, \frac{N^{2\tilde{T}} |\Theta| \tilde{T}^2 \log(N)}{K} \right\}. \quad (3.6)$$

This bound holds simultaneously for all possible reward functions bounded in $[0, 1]$. If the reward function r is constrained to be non-zero only for the terminal states $\mathcal{H}_{\tilde{T}}$, then

$$J(\pi^*) - \mathbf{E}_{\mathcal{D}} [J(\hat{\pi}_{\tilde{T}}^1(h; \theta, \mathcal{D}))] = \mathbf{E}_{p_{\tilde{T}}} [r(h_{\tilde{T}})] - \mathbf{E}_{\hat{p}_{\tilde{T}}} [r(h_{\tilde{T}})], \quad (3.7)$$

[18] using the definition of the marginal strategy $\hat{\pi}_{\tilde{T}}^1$. Finally, applying Lemma 1 of [73] gives

$$\|p_{\tilde{T}} - \hat{p}_{\tilde{T}}\|_{\text{TV}} \leq \min \left\{ \tilde{T}, \frac{N^{2\tilde{T}} |\Theta| \tilde{T}^2 \log(N)}{K} \right\}, \quad (3.8)$$

the desired result. \square

The imitation strategy $\hat{\pi}_{\tilde{T}}^1(h; \mathcal{D})$ marginalizes over agent 1's private type, and so the AI does not need to know its own type.

Imitate-then-commit strategy. For history $h_{\tilde{T}} \in \mathcal{H}_{\tilde{T}}$, let $\hat{z}(h_{\tilde{T}}) \in \Delta(N \times N)$ denote the empirical *joint* strategy played up to and including step \tilde{T} . Given $\hat{z}(h_{\tilde{T}})$, it is possible to construct a *mixture* ν over mixed strategies $x \in \Delta(N)$ such that, in expectation over ν , the partner's payoff under their best response to $x \sim \nu$ will be at least as large as their payoff under $\hat{z}(h_{\tilde{T}})$. The IC strategy described in Algorithm 1 follows $\hat{\pi}_{\tilde{T}}^1(h; \mathcal{D})$ for the first \tilde{T} steps, and then commits to a mixed strategy x for the remainder of the interaction. Then the following upper bound can be proved on the altruistic regret achievable with an imitate-then-commit strategy:

Theorem 3.6. *Given that Assumption 3.1 holds for ρ , if the AI follows $\pi^{IC}(\mathcal{D})$ (Algorithm 1) as agent 1, its altruistic regret satisfies*

$$E \left[\frac{1}{T} R_1^{alt}(h_T, \theta_2) \right] \leq \delta(K) + \epsilon_1 + \delta_1 + \frac{T - \tilde{T}}{T} (\epsilon_0 + \delta_0), \quad (3.9)$$

where $K = |\mathcal{D}|$ and $\delta(K)$ is defined as

$$\delta(K) = \min \left\{ 1, \frac{N^{2(\tilde{T}+1)} \tilde{T} \log(K)}{K} \right\} \quad (3.10)$$

[18] and where the expectation is taken over h_T , θ , and \mathcal{D} .

Proof of Theorem 3.6 First, let $\tau^2(\theta)$, defined as

$$\tau^2(\theta) = \min_{(\sigma^1, \sigma^2) \in \mathcal{P}(G(\theta))} G(\sigma^2, \sigma^1; \theta^2), \quad (3.11)$$

[18] denote agent 2's payoff under the worst possible payoff for a PONE of the game parameterized by joint type θ . Let \mathcal{C} denote the event that

$$\tau^2(\theta) - \frac{1}{\tilde{T}} \sum_{t=1}^{\tilde{T}} G(a_t^2, a_t^1; \theta_2) \leq \epsilon \quad (3.12)$$

Because ρ is $(\delta, \epsilon, \tilde{T})$ -compatible, then $\Pr_\rho\{\mathcal{C}\} \geq 1 - \delta$. For $\delta(K)$ defined as

$$\delta(K) = \min \left\{ \tilde{T}, \frac{N^{2(\tilde{T}+1)} |\Theta| \tilde{T}^2 \log(K)}{K} \right\}, \quad (3.13)$$

Lemma 5.2 also gives $\Pr_{\hat{\pi}^1, \rho}\{C\} \geq 1 - \delta - \delta(K)$. Therefore,

$$\mathbf{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=1}^T G(a_t^2, a_t^1; \theta_2) \right] \geq \mathbf{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=1}^T G(a_t^2, a_t^1; \theta_2) | C \right] - T(\delta + \delta(K)) \quad (3.14)$$

$$= \mathbf{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=1}^{\tilde{T}} G(a_t^2, a_t^1; \theta_2) | C \right] + \mathbf{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=\tilde{T}+1}^T G(a_t^2, a_t^1; \theta_2) | C \right] - T(\delta + \delta(K)) \quad (3.15)$$

$$\geq \mathbf{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=\tilde{T}+1}^T G(a_t^2, a_t^1; \theta_2) | C \right] + T(\tau^2(\theta) - \epsilon - \delta - \delta(K)) \quad (3.16)$$

This term can have a lower-bound:

$$\mathbf{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=\tilde{T}+1}^T G(a_t^2, a_t^1; \theta_2) | C \right] \quad (3.17)$$

This will be the expected payoff given the strategy $x \sim \nu$ the AI agent commits to for the remaining $T - \tilde{T}$ steps. The idea now is that a mixture ν over strategies can be constructed such that the the AI agent can commit to for the remaining $T - \tilde{T}$ steps such that the partner's payoff under their (approximate) best-response will be nearly as good as that under $\hat{z}(h_{\tilde{T}})$.

Let $G(z; \theta^2) = \sum_{i \in M} \sum_{j \in M} z_{i,j} G(j, i; \theta^2)$ be agent 2's expected payoff under z . For any joint strategy z , ν can be constructed such that if the AI agent commits to strategies sampled from ν , the partner will have the same information about the AI's probably actions as they would given their "recommended" action under $\hat{z}(h_{\tilde{T}})$. The construction used by Von Stengel and Zamir [312] motivate my work. For any joint strategy z , let $z_j = \sum_{i \in N} z_{i,j}$ denote the *marginal* probability that the column player (agent 2) plays j under z . For all $j \in N$ such that $z_j > 0$, x_j is defined as the *conditional* distribution over the row-player (agent 1's) actions given that the column player plays j , such that $x_j(i) = \frac{z_{i,j}}{z_j}$. Then ν is defined as the strategy that commits to each x_j with

probability z_j .

It is shown that when the partner plays a best response to $x \sim \nu$, their payoff will be no worse than under z itself. At first, a *response function* r_z is constructed such that when agent 2 responds to $x \sim \nu$ with $r_z(x)$, its expected payoff equals $G(z; \theta^2)$. Let $S = \{j \in N : z_j > 0\}$, and partition S into \mathcal{P} such that, for each $P \in \mathcal{P}$, there is a $x_j = x_l$ for all $j, l \in P$. For each $P \in \mathcal{P}$, the strategy y_P can be defined such that

$$y_P(j) = \frac{z_j}{\sum_{l \in P} z_l} \quad (3.18)$$

for each $j \in P$, with $y_P(j) = 0$ for $j \notin P$. (Note that if z corresponds to some *uncorrelated* strategy $\langle x, y \rangle$, then $P = N$ and $y_P = y$.) Finally, for $j \in S$, $P(j)$ is defined as the partition containing j , and define r_z such that $r_z(x_j) = x_{P(j)}$. r_z can be left undefined for x where $\mu(x) = 0$. Now let x_P be the common conditional strategy for all $j \in P$, and let $z_P = \sum_{j \in P} z_j$. Then,

$$\mathbf{E}_{x \sim \nu} G(r_z(x), x; \theta^2) = \sum_{j \in S} z_j [x_j^\top G(\theta^2)^\top r_z(x_j)] \quad (3.19)$$

$$= \sum_{P \in \mathcal{P}} z_P [x_P^\top G(\theta^2)^\top y_P] \quad (3.20)$$

$$= \sum_{P \in \mathcal{P}} z_P \left(\sum_{i \in N} \sum_{j \in N} x_P^\top(i) y_P(j) G(\theta^2)_{ij}^\top \right) \quad (3.21)$$

$$= \sum_{P \in \mathcal{P}} z_P \left(\sum_{i \in N} \sum_{j \in N} \Pr_z\{i|P\} \Pr_z\{j|P\} G(\theta^2)_{ij}^\top \right) \quad (3.22)$$

$$= \sum_{P \in \mathcal{P}} z_P \left(\sum_{i \in N} \sum_{j \in N} \Pr_z\{i, j|P\} G(\theta^2)_{ij}^\top \right) \quad (3.23)$$

$$= \sum_{i \in N} \sum_{j \in N} z_{ij} G(\theta^2)_{ij}^\top = G(z; \theta^2) \quad (3.24)$$

the above holds using the fact that i and j are independent given that $j \in P$. Next, for any best-response function r^* , there is

$$\begin{aligned} G(z; \theta^2) &= \mathbf{E}_{x \sim \nu} G(r_z(x), x; \theta^2) \\ &= \mathbf{E}_{x \sim \mu} [x^\top G(\theta^2)^\top r_z(x)] \\ &\leq \mathbf{E}_{x \sim \mu} \left[\max_{y \in \Delta(N)} x^\top G(\theta^2)^\top y \right] \\ &= \mathbf{E}_{x \sim \mu} [x^\top G(\theta^2)^\top r^*(x)] \\ &= \mathbf{E}_{x \sim \nu} G(r^*(x), x; \theta^2) \end{aligned} \quad (3.25)$$

Therefore, so long as the partner plays a best-response to the AI agent's chosen strategy, they will achieve at least the same payoff (in expectation) as they would under the strategy z

from which ν was computed. Note however that ρ will be (approximately) consistent over the full T steps, not just the last $T - \tilde{T}$. Define $\alpha = \frac{\tilde{T}}{T}$ and $\beta = \frac{T - \tilde{T}}{T}$, and let z^1 be agent 1's marginal strategy under z . With probability $1 - \delta$, ρ will play an ϵ -best-response to the mixture $\alpha \hat{z}(h_{\tilde{T}})^1 - \beta x$, with $x \sim \nu$.

Let \mathcal{C}' be the event that ρ is ϵ -consistent over T steps. Then,

$$\mathbb{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=\tilde{T}+1}^T G(a_t^2, a_t^1; \theta_2) | \mathcal{C} \right] \geq \mathbb{E}_{\hat{\pi}^1, \rho} \left[\sum_{t=\tilde{T}+1}^T G(a_t^2, a_t^1; \theta_2) | \mathcal{C}, \mathcal{C}' \right] - T\delta \quad (3.26)$$

$$\geq (T - \tilde{T}) (\tau^2(\theta) - 2\epsilon) - T\delta \quad (3.27)$$

Finally, dividing by T and subtracting from $\tau^2(\theta)$, the desired result

$$\mathbb{E} [R^{\text{alt}_1}(h_T, \theta_2)] \leq 2\delta + \delta(K) + \left(2\frac{T - \tilde{T}}{T} + 1 \right) \epsilon \quad (3.28)$$

can be derived. \square

3.7 Use Cases of Socially Intelligent Agents

3.7.1 AI Health Coach

Imagine a health app that introduces an AI assistant designed to help users with behavioral change in planning their fitness routines while balancing workload or classes as students. The AI agent is trained using data from prior interactions between human coaches and users, aiming to replicate the guidance and intervention typically offered by fitness coach professionals. However, a model based purely on imitation learning struggles to maintain long-term engagement. While

the AI agent can provide helpful advice in the short term, it fails to adapt to the unique preferences and behavioral tendencies of individual users. For example, the AI agent might overlook that one user prefers mindfulness exercises in the mornings, or that another prefers structured goal-setting rather than open-ended reflection.

The proposed approach is to first imitate the human coach’s guidance over a short time horizon. During this phase, the AI agent gathers enough data to identify the user’s latent preferences, routines, and physical features to match with the intensity of their fitness routines (their ‘type’). After this initial phase, the AI agent transitions to a type-conditioned policy, where it tailors its suggestions to better fit the user’s specific needs—such as recommending mindfulness sessions at optimal times or adjusting goals based on prior feedback. This prevents early disengagement, where users might otherwise feel that the AI agent is offering generic advice that doesn’t resonate with their personal behavioral patterns, leading them to discontinue its use.

3.7.2 Human-Robot Cooperation in a Factory

As an illustrative example, consider the problem of designing a control policy for a cooperative robot intended to assist a human in a manufacturing factory. The challenge is that, while the AI will know the basic steps on how to lift things or how to weld metal, the human engineer or factory will have distinct preferences about the planning and physical reasoning aspects in a factory like which box is to be moved and how in case it has high value goods. The human worker can plan how to divide the task between themselves and the robot so that the robot can help in welding metal using the design provided by the human and then the robot will move the manufactured goods. The expectation is that the robot will adapt to the human worker’s prefer-

ences over time, but there are many ways these might be communicated. One worker may ignore the robot, expecting it to take over tasks as it is able to do so, while another might describe their preferences verbally. To learn an adaptive policy, past observations of human worker populations can be leveraged for learning to work together for the first time. From these observations, strategies can be learned that the "population" of human workers tend to use when adapting to a new partner.

This practical application can be accordingly mapped to the formal model of Social Intelligence. Each partner agent drawn from the population is an individual worker, and each agent's (worker's) preferences for how items should be manufactured and moved are encoded in its "type". The "stage game" corresponds to the manufacturing of a single item, such that pure strategies in this game correspond to (short-term) policies to be executed in the factory manufacturing environment. Each episode describes the entire period the robot spends interacting with a single human worker, and both the robot and its partner can condition their behavior on the entire history of this interaction, that is, the history of attempts to manufacture and move items together. The population-level dataset would consist of multiple complete histories of interactions between human workers, that show not only the strategies they use to manufacture items, but the strategies they use to convey their private preferences (types).

3.8 Related Works

This work is closely related to the previous targeted learning model [57, 239, 240], which defines similar compatibility and consistency criteria. The notion of targeted optimality [57] includes convergence to learning an approximately best response in a Multi-Agent model with

high probability in a tractable number of steps against a population of memory-bounded adaptive agents. The main difference with my Social Intelligence work is that targeted learning only requires consistency against a specific target class of partners, which generally would not include the agent itself, or other adaptive agents. Socially intelligent agents need to be consistent against all possible partner strategies. Cooperation and consistent learning also have to occur over a fixed time horizon T , rather than asymptotically. These differences mean that a hypothetical “universally cooperative” [18] agent can possibly leverage the consistency of its partner to achieve cooperation without a prearranged convention. “Universal cooperation” in a population is relevant for specific populations as described in the illustrative examples like surgeons in a hospital, workers in a factory etc. In these settings it is reasonable to assume that most agents will be able to cooperate with each other professionally, since if a member fails to do this, they would not be a member of that population. Furthermore, my formal model allows agents in the population to possess highly conflicting preferences with my definition of cooperation only requiring agents to identify mutually beneficial joint strategies when these exist. Socially intelligent agents can be modeled as individually rational learners [189] to achieve Pareto-efficient joint behavior. Recent research explores the problem of bounding the behavior of the socially rational agents but ends up with lower bounds due to additional assumptions on the private type of the population, while it is unknown to the AI Agent [186] which also make it difficult to be extensible to different imitation learning problems. My research builds on these works in a learning setting where the agent when paired with any member of the population will achieve at least the same utility with high probability as the Pareto-efficient approach. In the real world, the sample complexity of my imitation learning and imitate-then-commit algorithms are better due to worst-case scenarios determining the bounds. Sample complexity of my bounds in the real world will also depend

a lot on the alignment of the function approximation of the imitate-then-commit strategy to the learning task. My sample complexity bounds can be applied to any imitation learning problem including inverse RL, behavioral cloning, etc. with relaxed compatibility assumptions on human population's socially intelligent behavior. My model considers the bounded rationality of humans as their private types are impacted by external environment and individual choices, including when the human agents in a population have varying skill levels. This is addressed by the altruistic regret definition penalizing skills of the specific human agent (e.g. a Doctor in Residency), not relevant to the task and allows the AI Agent to adapt to the skills of their individual collaborating partner.

The problem of training agents to be able to cooperate with previously unseen partners is sometimes referred to as *ad hoc teamwork* [213, 283] or *zero-shot coordination* [138], especially in the context of Multi-Agent reinforcement learning. Many approaches in reinforcement learning train cooperative policies that are *robust* to possible strategies that a human or an AI agent can follow [53]. A lot of these methods build a “population” of partner strategies and maximize the diversity of this population in order to train the AI's policy against it [80, 284]. Other approaches assume that there is no prior coordination between the agents [138] to learn rational joint strategies while estimating the agents' mutual uncertainty about one-another's strategies [304]. Ad-hoc Multi-Agent coordination can be helpful to learn cooperation among AI agents with the “other-play” algorithm [138] that finds such a strategy as a solution to the corresponding *label free coordination* problem [304]. A possible approach to solve these problems can be self-play [345] where the agent can optimize themselves by playing with past iterations of themselves in order to estimate the strategies of unseen partners. However, the "self-play" approach can learn cooperative strategies which can "over-fit" [284] to one another in the population of

agents. A key goal of Ad hoc coordination (teamwork) and aligned research in zero-shot coordination work has been to avoid this type of overfitting [79]. This problem domain is closely related to both *ad hoc teamwork* or *zero-shot coordination*, since training an AI agent to cooperate with previously unseen partners with no control over the partner is considered. Even though population-based training approaches to ad hoc teamwork are common, they focus on fully cooperative environments such as Dec-POMDPs, where the main issue is creating a diverse enough population to train with [244]. Only self-interested partners who do not assume identical payoffs are considered.

Finally, in the case of Hannan-consistent partners, this problem setting is closely related to strategizing against and learning to manipulate no-regret learners [44, 87]. This line of work studies whether an optimizer agent can achieve better payoff than CCE against no-regret learners by learning to enforce a Stackelberg equilibria on them. Their emphasis is on online learning and the optimizer’s payoff, while my work is on the offline setting and cooperation.

3.9 Conclusion

Formal guarantees for successful and reliable cooperation of AI agents with populations of socially intelligent agents are achieved. A novel definition of social intelligent populations is based on the assumptions that 1) members of the population are individually rational, and 2) pairs of members can achieve performance comparable to a Pareto-optimal Nash equilibrium. The notion of consistency and compatibility of agents is formalized in repeated, two-player, general-sum matrix games with private types. The theoretical guarantees are in the offline cooperation setting where the agent has to cooperate with unseen partners in the population to strategies

against and manipulate no-regret policies for which the idea of altruistic regret is formalized. It is then proven that the assumptions on its own are insufficient to learn *zero-shot* cooperation with partners of the socially intelligent target population. Upper bounds on the sample complexity needed to learn a successful cooperation strategy are provided along with lower bounds on when the Multi-Agent cooperation setting is needed with respect to the populations' trajectories, the state space and the length of the learning episodes. The bounds in these settings of the agent actively querying the MDP without knowing the transition dynamics of the population or the agent observing the populations' transition dynamics are much stronger than the bounds that can be derived by naively reducing the cooperation problem to one of reinforcement learning. These complexity analyses and formally proven bounds can be helpful to sustainably model the alignment problem of AI agents.

Chapter 4: Social Cooperation in Conversational AI Agents

General-purpose AI assistants are capable of supporting people in a wide variety of tasks, including writing, graphic design, coding, and scientific research. The development of these AI assistants has been facilitated by the introduction of AI agents based on open-domain Large Language Models (LLMs). Such agents are trained by observing relatively short-term interactions with humans, posing a major challenge. Models routinely fail at generalizing to long-term interactions, such as the agent being repeatedly told to correct a mistake it makes by a user. In this chapter, the aim is to give theoretical bounds that address these challenges with the explicit modeling of *social intelligence*, that is, the ability of humans to maintain and create long-term relationships, even when their behavior cannot be predicted. Using a mathematical modeling of how humans reason and communicate about one another over a long time, *game theoretic* objectives can be derived. These allow for future AI agents and LLMs to be optimized for sustained cooperation.

4.1 Introduction

The ability to sustain long-term interactions is needed in order for conversational AI assistants [133, 254] to effectively collaborate with human users while also adapting to their individual needs. Consider a hypothetical example where Alice, a team lead at a software firm,

introduces an AI assistant into her daily workflow. Alice spends time teaching the AI assistant her empathetic and constructive, rather than blunt, style of communication. Over time, she notices subtle but very important issues. The AI assistant gives impersonal, terse feedback in high-pressure situations despite repeated guidance to behave otherwise. It also allocates time unpredictably, without consideration for the personal constraints of colleagues. Alice's team responds by no longer telling her about their constraints, and the AI assistant does not ask for them. Ultimately, Alice removes the AI assistant [18].

Suppose that the AI assistant we described above is fine-tuned or pretrained on a dataset of conversations between employees at similar workplaces. Then our earlier example presents two major technical challenges for creating the AI assistant. One challenge is using long conversation histories to learn how to act appropriately and using prior guidance to adjust, in addition to appropriately managing workloads [305]. The second challenge is that small differences between how a human assistant would behave and the actions of an AI assistant can make the users act much more differently than they would otherwise (such as intentionally withholding information), leading to a gap between the training data and real-world interactions.

We examine whether these issues can be mended by assuming that human users do not behave arbitrarily. In particular, we have two assumptions that we formalize. The first is that over time, human users will gradually adapt to any agent they are speaking with, whether they are human or AI. Second, that populations of human users follow certain "conventions" that enable them to collaborate with new partners. We model this problem by defining the notion of "social intelligence" for agents that satisfy these constraints, and prove that strategies for populations of these agents to collaborate can be learned efficiently. Our theoretical results suggest that traditional fine-tuning or imitation learning objectives may be suboptimal when training AI agents

to collaborate with human users.

4.2 Preliminaries

Long-term collaborations between members of a target population are modeled as *repeated two-player games with private utilities*, in keeping with the conventions established in Section 2.1.6.

4.3 Social Intelligence

For our framework of social cooperation for conversational AI agents, we assume the agents to be both *consistent* and *compatible*, adhering to definitions 2.1 and 2.2

To see the effects of social intelligence for communication, consider the case when a cautious employee is paired with an employee who needs and values directness. If they belong to the target population \mathcal{C} , they should be able to identify each other's types and adapt their communication strategies in order to avoid individual regret and achieve approximate compatibility. Our goal is to select a meta-strategy for the AI agent that can communicate and cooperate with a partner picked from some *target population* nearly as effectively as agents from this population do so with one another. For the class of games $\mathcal{G} = (\mathcal{I}, \mathcal{A}, \Theta, G, T)$, we will let the target population be a set \mathcal{C} of agents forming a (δ, ϵ, T) -SI class with respect to Θ [18].

The goal is to have an AI meta-strategy π capable of cooperating with \mathcal{C} *without* any further knowledge of the strategies in \mathcal{C} . This means analyzing the problem of using prior observations of members in the target population interacting with one another to learn a cooperative meta-

strategy. A tuple $\{\mathcal{G}, \mathcal{C}, \rho, \mu\}$ defines a *social learning problem*, where \mathcal{C} is the target population (SI w.r.t. Θ), ρ is a distribution over the members of \mathcal{C} , and μ is a distribution over $\Theta \times \Theta$, the joint type space. \mathcal{C} is akin to the set of possible strategies that any member of the target population might follow, while ρ is how frequently each strategy type in \mathcal{C} appears. A dataset $\mathcal{D} = \{(\theta_1^j, \theta_2^j, h_T^j) | j \in [n]\}$ covering n episodes of length T is used to choose an AI strategy. In each episode j , two agents π_j^1 and π_j^2 are sampled independently from ρ , and played together under the joint type $\theta^j \sim \mu$. The AI agent observes the full history h_T^j , along with the agents' types θ_1^j and θ_2^j . A specific learning algorithm is denoted as a data conditioned strategy $\pi(\mathcal{D})$.

Our objective is to encourage cooperation through minimizing Altruistic Regret, defined in Definition 3.1. The agent must know its own type as well, or the user agent may switch to a safe strategy like disengaging the AI agent, as described in the introductory example.

4.3.1 Theoretical Ablations

It is not yet obvious that both consistency and compatibility are required for the populations that can be modeled naturally through this definition of social intelligence. If it is assumed the agents in \mathcal{C} have strategies which are consistent but may not be compatible, then the most general class is the case of all no-external-regret-learners which we call no-regret learners. In the long run, these no-regret learners average out to the set of coarse-correlated equilibria (CCE). Then our question becomes whether it is possible for the AI agent to learn a better strategy than a CCE when partnered with a member of \mathcal{C} , given a dataset \mathcal{D} of histories for game players under different CCE.

Unfortunately, learning in the case of consistency alone cannot be guaranteed. It is shown

in theorem 3 of [217] that given any coarse correlated equilibrium of a two-player normal-form game, a pair of no-regret learners exists that would converge to it. We cannot exclude those who converge to inefficient CCE since \mathcal{C} can be any set of no-regret learners. If the class \mathcal{C} includes only the agents that converge to Pareto-inefficient CCE, we cannot hope to learn optimal strategies from any dataset with respect to the altruistic regret. Additionally, regardless of the dataset, when the AI's strategy is deployed, it will face a partner drawn from \mathcal{C} with an unknown type. We may wish to learn a classifier to quickly infer our partner's type online from their behavior, assuming a mapping from initial behavior to types exists. However, since \mathcal{C} includes only no-regret learners guaranteed to converge to a CCE in self-play, they have no reason to communicate their types to each other.

Now let us consider that the members of \mathcal{C} are compatible but not consistent. Suppose that the members of \mathcal{C} have an established handshake protocol, which allows them to communicate and identify each other's types at the start of their interaction. When they identify each other after the handshake phase, they proceed with playing the agreed-upon PONE. However, if at any moment the wrong action is played by one agent, the strategy the other will switch to is unconstrained. Grim-trigger strategies can be employed by the members of \mathcal{C} , punishing the other agent for the rest of the horizon, triggered by a mistake at any point. The finite-horizon setting signifies that even with a forgiving strategy, mistakes can in effect lead to grim-trigger outcomes. Then the outsider must learn to imitate at least one member of \mathcal{C} perfectly from the dataset. The offline problem in this setting reduces to imitation learning, in particular the no-interaction case from [245]. Unfortunately, the sample complexity of the no-interaction imitation learning scales exponentially with horizon in our case, since the strategies are history-dependent [18].

4.3.2 Upper Bound for Social Cooperation

In this Chapter, a central concept is that an AI agent only needs to imitate a member of a socially intelligent population just long enough to learn the private type of its partner, rather than trying to copy them throughout the entire episode. Knowing this, the agent can recall that the strategy of its partner is consistent against *any* other strategy. This means the agent can try to "incentivize" its partnered human to adopt a strategy that minimizes altruistic regret. These strategies, where the AI agent selectively imitates their partner, can be called *imitate-then-commit* (IC) strategies. Formally, IC strategies use the set of prior observations \mathcal{D} to learn their imitation strategy, which they follow for the first $\tilde{T} < T$ steps of the episode. The Altruistic Regret of a specific (IC) strategy, as a function of the number of episodes in \mathcal{D} , has an upper bound when subject to certain assumptions [18], which are shown in this section:

Assumption 4.1. For $\delta, \epsilon > 0$, and some $\tilde{T} < T$, we have that (1) ρ is (δ, ϵ, T) -consistent, and (2) ρ is $(\delta, \epsilon, \tilde{T})$ -compatible.

Imitation learning. Under an imitate-then-commit strategy, the number of episodes the AI agent needs to observe to learn a good \tilde{T} -step imitation policy entirely defines the sample complexity. Fortunately, imitation learning is a well-studied problem, and existing complexity bounds can be leveraged. The one caveat is that in this setting bounds on the total variation distance between the distribution over the partial history $h_{\tilde{T}}$ under the population strategy ρ , and that under the learned strategy are needed. Given the dataset \mathcal{D} , we define the imitation strategy $\hat{\pi}_{\tilde{T}}^1(\mathcal{D})$ such that $\hat{\pi}_{\tilde{T}}^1(h; \theta, \mathcal{D})$ is the empirical distribution over agent 1's actions for each history-type pair (h, θ) occurring in \mathcal{D} , while $\hat{\pi}_{\tilde{T}}^1(h; \theta, \mathcal{D})$ is the uniform distribution over N for $(h, \theta) \notin \mathcal{D}$.

We then define the *marginal* strategy $\hat{\pi}_{\tilde{T}}^1$, which can be implemented by sampling a dataset \mathcal{D} , and then following the imitation strategy defined by \mathcal{D} for the next \tilde{T} steps. We then have the following bound on the distribution of $h_{\tilde{T}}$ under the imitation strategy:

Lemma 4.1. *Let $p_{\tilde{T}}$ be the distribution over partial histories $h_{\tilde{T}}$ under the population strategy ρ , and let $\hat{p}_{\tilde{T}}$ be their distribution under $\hat{\pi}_{\tilde{T}}^1$. We have that*

$$\|p_{\tilde{T}} - \hat{p}_{\tilde{T}}\|_{TV} \leq \min \left\{ \tilde{T}, \frac{N^{2(\tilde{T}+1)} |\Theta| \tilde{T}^2 \log(K)}{K} \right\}, \quad \text{where } K = |\mathcal{D}|. \quad (4.1)$$

This bound follows directly from that of [245] via Lemma 1 of [73] (which follows from the detailed proof in Section 3).

Imitate-then-commit strategy. For history $h_{\tilde{T}} \in \mathcal{H}_{\tilde{T}}$, let $\hat{z}(h_{\tilde{T}}) \in \Delta(N \times N)$ denote the empirical *joint* strategy played up to and including step \tilde{T} . We show that, using $\hat{z}(h_{\tilde{T}})$, it is possible to construct a *mixture* ν over mixed strategies $x \in \Delta(N)$ that, in expectation over ν , the partner's payoff under their best response to $x \sim \nu$ will be at least as large as their payoff under $\hat{z}(h_{\tilde{T}})$. The corresponding IC strategy will operate as follows:

1. Sample \mathcal{D} and compute the imitation strategy $\hat{\pi}_{\tilde{T}}^1(\mathcal{D})$.
2. Play $\hat{\pi}_{\tilde{T}}^1(\mathcal{D})$ for the first \tilde{T} steps, and observe $h_{\tilde{T}}$.
3. Compute a suitable mixture ν from $\hat{z}(h_{\tilde{T}})$, and sample $x \sim \nu$.
4. Sample actions from x for the remaining $T - \tilde{T}$ steps.

We then have the following upper bound on the altruistic regret achievable with an imitate-then-commit strategy:

Theorem 4.2. *Given that Assumption 4.1 holds for ρ , there exists a data-dependent strategy $\pi^{IC}(\mathcal{D})$ such that when played by the AI as agent 2, the altruistic regret satisfies*

$$E [R_1^{alt}(h_T, \theta_2)] \leq 2\delta + \delta(K) + \left(2\frac{T - \tilde{T}}{T} + 1\right) \epsilon, \quad \delta(K) = \min \left\{ \tilde{T}, \frac{N^{2(\tilde{T}+1)} |\Theta| \tilde{T}^2 \log(K)}{K} \right\} \quad (4.2)$$

where $K = |\mathcal{D}|$ and the expectation is taken over h_T , θ , and \mathcal{D} .

Proof sketch: By Lemma 4.1, we can learn an imitation strategy such that the corresponding distribution over $h_{\tilde{T}}$ and $\hat{z}(h_{\tilde{T}})$ is close to that under ρ in self-play. As ρ is compatible, both agents' payoffs under $\hat{z}(h_{\tilde{T}})$ must be close to those under *some* PONE. Finally, we can construct a mixture ν for agent 1 such that agent 2's payoffs under its (approximate) best-response are almost as large as those under $\hat{z}(h_{\tilde{T}})$ (the detailed proof flows from the theoretical proofs in Section 3) [18].

4.4 Conclusion

It is a challenging problem to train conversational AI agents in order to maintain long-term interactions and cooperation with human partners. Naive reduction of this problem to imitation learning leads to infeasibility, due to the statistical limitation with large state spaces and long horizons. Instead, formal guarantees are provided for successful and reliable cooperation of AI agents with populations of socially intelligent agents. A novel definition of socially intelligent populations is presented based on the assumptions that 1) members of the population are individually rational, and 2) pairs of members can achieve performance comparable to a Pareto-optimal Nash equilibrium. The notion of consistency and compatibility of agents are formalized in re-

peated, two-player, general-sum matrix games with private types.

In an offline cooperation setting our theoretical guarantees apply where the agent must strategize and manipulate no-regret policies to cooperate with unseen partners. For these situations our formalization of altruistic regret is useful. Our upper bounds constrain the sample complexity needed to learn a successful cooperation strategy. These formal bounds are useful to model the AI agent alignment problem in a sustainable manner.

Chapter 5: Multimodal Vision-Language Proactive AI agents for Augmented Reality

Multimodal AI Agents are AI models which can interact, cooperate and assist human users to solve day-to-day tasks. Augmented Reality (AR) head worn devices uniquely improve the user experience by giving AI Agents multimodal AR capabilities of audio and video to observe the users' actions. Existing AI Agents, either LLMs or VLMs are reactive which cannot take an action without receiving any human input. Proactivity of AI Agents provide additional autonomy to help the human user to detect and correct any mistakes for real-time tasks, motivate users when they do tasks correctly or simply chat with users. My proposed YET to Intervene (YETI) multimodal AI agent [17] focuses on the research question to detect circumstances that may require it to intervene proactively. This allows the agent to understand when it can intervene in a conversation with human users e.g. to help users fix mistakes in cooking, using AR. The YETI Agent learns a scene understanding signal to get a semantic understanding of the environment around the user, using a form of Structural Similarity (SSIM) between two consecutive observed video frames. It also uses an alignment signal to determine if the video frames corresponding to the user's actions on the task are actually showing that users are following expected actions. Proactive intervention results of YETI Agent are compared to the baselines of the HoloAssist multimodal benchmark for an expert agent guiding a user to complete procedural tasks.

5.1 Introduction

With rapid and widespread deployment of Artificial Intelligence (AI) Assistants be it speech-interaction based Siri or vision-speech-text based Gemini [297], there is a growing necessity to have agency in AI models. AI Agents act and guide humans for completing real-world tasks like cooking or electric maintenance. These AI Agents can interact with humans using speech, video or text through hand-held phones or emerging wearable Augmented Reality (AR) devices like glasses or headsets, presenting a new paradigm of human-AI interaction. This is especially useful in scenarios that require real-time, context-aware assistance.

Multimodal Vision-Language Models (VLMs) are a promising solution for AI Agents to have multimodal understanding similar to human perception, letting them anticipate human needs. However, current VLM-based assistive systems operate in a reactive paradigm, responding only to explicit user queries. This reactive nature limits the usefulness of AI Agents significantly in two critical scenarios:

1. **Novice learning environments:** Users lack domain knowledge to form relevant queries like in education.
2. **Safety-critical operations:** AI Agents may need to immediately guide the user around potential hazards.

An example of the first scenario might be a student doing homework. Current AI models tell the right or wrong answer without improving the student's problem-solving skills. The second scenario could occur in a manufacturing environment where a worker handles seemingly safe pieces of metal without noticing loose electric wires. Here, a reactive AI Agent cannot help, but

a proactive AI Agent can. A wearable AR device (like a glass) gives the AI Agent visibility of the worker’s view and guides them about the hazard they were unaware of.

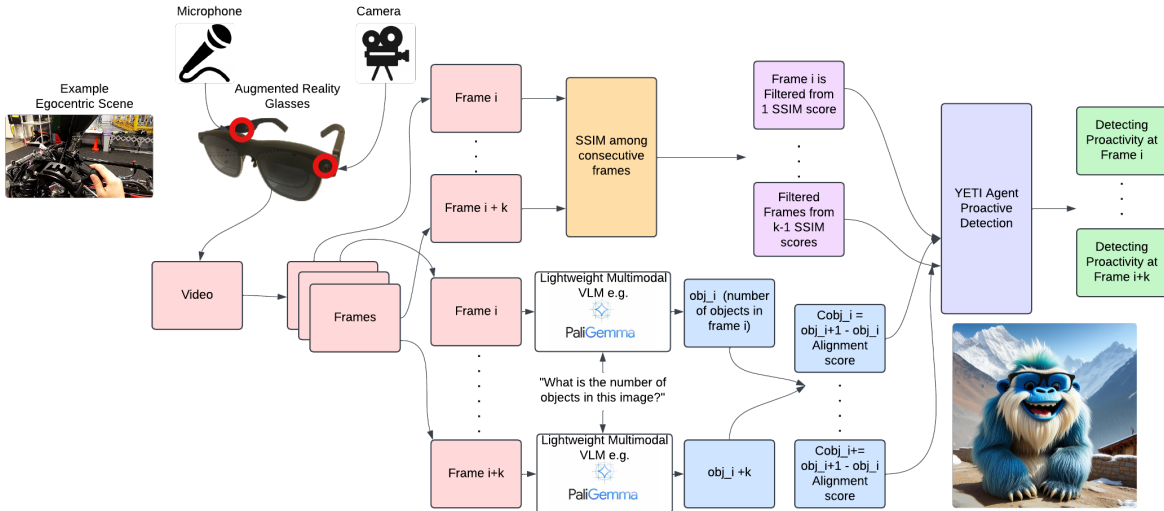


Figure 5.1: Overview of the YETI framework for detecting proactive interactions or interventions in Augmented Reality. YETI uses lightweight signals generated in real-time, allowing timely decision making for providing assistance to the user [17].

To address these challenges, I propose a novel framework **YET** to **I**ntervene (YETI) for proactive AI Agents to intervene in AR environments. Proactive interventions allow AI to decide when to help the user without waiting for the user’s question. This approach uses lightweight, real-time algorithmic signals to detect when proactive assistance is needed through AR interfaces such as smart glasses [90, 315]. The system bridges the gap between cloud-based AI capabilities and real-world applications with AI that sees direct visual observations of user activities in the physical world. My work is motivated from recent developments in proactive AI assistance, particularly the HoloAssist dataset [319], which demonstrates the potential for real-time AI intervention in complex procedural tasks. Valuable insights are provided by HoloAssist into human-AI collaboration scenarios, such as computer assembly and coffee preparation, but existing implementations face significant computational challenges.

Current state-of-the-art approaches for proactive intervention require extensive computational resources and multimodal sensor data that includes RGB streams, hand and head pose estimation, sensor readings like IMU (Inertial Measurement Unit), and depth maps. A significant barrier to practical deployment is the complexity of acquiring and processing this data in real-time. In contrast, YETI employs efficient algorithmic signals computed on-the-fly that dramatically reduce the computational overhead while maintaining high intervention performance.

Features	Size (MB)	× SSIM	× CObj
Depth Estimation	137,408	6,543	6,870
Eye Gaze (E)	617	29	31
Hand Pose (H)	53,749	2,660	2,688
Head Pose	1,141	54	57
IMU	1,132	54	57
SSIM(Ours)	21		
Alignment_Cobj(Ours)	20		

Table 5.1: HoloAssist Feature sizes scaled with YETI Features

These contributions can be summarized as follows:

1. YETI, a novel AI Agent framework for real-time proactive intervention in AR environments.
2. Efficient algorithmic signals based on Structural Similarity (SSIM) and changing object count (Alignment_CObj) that can be computed online, reducing computational overhead by as much as $6,870\times$ (seen in Table 5.1) compared to HoloAssist.
3. Superior recall and F-measure compared to HoloAssist baselines while maintaining competitive precision.
4. Ablation studies to show the most critical components of this approach, giving insights to

developers of future assistive AI systems.

5.2 Related Works

5.2.1 Egocentric Interaction Datasets

Recent advances in egocentric vision have produced several datasets which contain reactive human-AI interactions. HoloAssist [319] is a large-scale egocentric proactive dataset that focuses on physical manipulation tasks between two people in a collaborative manner, providing detailed action and conversational annotations. Valuable insights into how human assistants proactively, as well as reactively intervene, correct mistakes, and ground their instructions in the environment are offered in this dataset.

Other benchmarks like Parse-Ego4D [2] evaluate the capability of AI agents to make unsolicited action suggestions based on user intent. This benchmark does not measure proactive behavior as it requires the AI agents' respond to user queries.

Bigger benchmarks like Ego-Exo4D [120] focus on skilled single-person activities without addressing proactive communication and provides a comprehensive multimodal, multiview dataset capturing both egocentric and exocentric perspectives in expert-learner scenarios. Similarly, existing datasets like Ego4D [119] and EPIC-Kitchens [82] are rich in activity and object annotations, but lack proactivity or direct mappings to actionable recommendations.

5.2.2 Proactive AI Agents and Communication

Proactive communication in AI agents includes several key aspects [86]: Intelligence (the ability to predict and act on task developments), Adaptivity (dynamic adjustment of location and timing of interventions), and Civility (respect for user boundaries and ethical standards). The value of proactive AI agents across various domains, include personalized assistance, predictive maintenance, healthcare monitoring, increasing economic and manufacturing innovation, education, and voice assistance [45, 46, 182].

5.2.3 Language Models for Proactive Assistance

Promising results are emerging with LLMs for proactive assistance. The ProAgent framework [347] leverages LLMs to create agents that dynamically adapt their behavior and infer teammate intentions. The effectiveness of these models has been further shown through fine-tuning on ProactiveBench [196], significantly improving the proactive capabilities of LLM agents. Smart Help [51] demonstrates how proactive and adaptive support, in the context of assistive technology, can be provided to users with diverse disabilities and dynamic goals across various tasks and environments. It is to be noted that the new research with proactive agents mostly focus on LLMs unlike my research on VLM-based proactive YETI agent.

Open-source VLMs, especially Google’s PaliGemma [33], are popular for AI assistants, with small-scale variants like Gemma-3 [298] being run on edge devices. Open-source VLMs do not have proactive interaction capabilities which I have supported in my research. PaliGemma generates a quick and accurate description including an estimate of the number of objects in a given scene. PaliGemma is trained on a wide variety of datasets, including the TallyQA

dataset [3], which is useful for taking a response to a question that asks for the number of objects in a given image.

5.2.4 Procedural Mistake Detection

The analysis of egocentric data has been explored in previous studies that explore assistance with procedural tasks; however, none have approached this challenge in the same comprehensive manner as YETI. YETI is designed to detect optimal moments for proactive intervention. On the other hand, existing works primarily focus on mistake detection, limiting direct comparisons with YETI’s broader intervention detection capabilities.

Online mistake detection is done in **PREGO** [101] (Mistake Detection in **PR**ocedural **EGO**centric Videos) in a similar manner to YETI’s real-time intervention detection. However, PREGO is confined to the Mistake Detection intervention type with just visual inputs. YETI looks at both vision and language inputs, exploring more scenarios outside mistake detection where AI intervention can be beneficial. Specifically, PREGO lacks mechanisms for proactive interventions, requiring users to make errors before the AI can respond. Although PREGO incorporates step anticipation, it only detects deviations from predefined plans to identify mistakes. While PREGO necessitates a symbolic description of the task for mistake identification, whereas YETI operates directly on video frames without such annotations.

The work of PREGO is extended in **TI-PREGO** [237], comprehensively integrating LLMs for action anticipation and detection modules along with incorporation of chain-of-thought reasoning and in-context learning. The ability of several LLMs (LLaMA, Mistral, Gemma, and GPT) to perform step anticipation is evaluated. TI-PREGO focuses solely on mistake detection

within procedural tasks and does not expand into proactive intervention detection.

Online mistake detection is also addressed in **Differentiable Task Graph Learning** [268] but diverges by utilizing Task Graphs instead of LLM-based symbolic reasoning. Task Graphs model procedures as sequences of steps with directed dependencies, ensuring certain steps precede others. However, pre-segmented key-step sequences from input videos are needed here, rendering it unsuitable for real-time Augmented Reality applications. On the other hand, YETI processes continuous actions without necessitating pre-annotated data, allowing for real-time operation.

The reliance on supervised learning observed in Differentiable Task Graph Learning is mitigated in **Eyes Wide Unshut** [206] by predicting mistakes based on eye gaze trajectories instead of manual annotations. The next gaze position of the user is forecasted by this method during task execution and compares it with actual gaze data to detect discrepancies. Unlike YETI, Eyes Wide Unshut heavily depends on the availability of eye gaze data, which is not always be accessible, specially for real-time AI Agent deployment on distributed edge computing devices. Furthermore, many tasks do not require significant gaze shifts that limit the method's applicability and increase the potential for false positives when gaze changes are unrelated to task performance errors.

5.2.5 Action Detection

Broad action detection has been explored by other works that classify actions detected in videos instead of detecting when to proactively intervene or when a user makes a mistake.

Quasi-Online Detection of Take and Release Actions [263] focuses on near real-time

detection that allows a slight delay between an action and its detection on-the-fly. This work focuses on identifying "take" and "release" actions, which are instances where there is an interaction of the user's hands with objects instead of focusing on mistake or intervention detection. While downstream tasks like intervention or mistake identification can be supported by such action detection with assumptions, it does not directly address these areas, thereby limiting its overall scope and applicability for use as an AI assistant.

One of the initial research to use Transformer-based models for online action detection is **OadTR** [320], pivoting from Recurrent Neural Networks (RNNs), which exhibit less parallelism. A key difference between OadTR and this work is the point-of-view of the videos and the latency of the AI Agent's response. My YETI agent uses video frames captured from a first-person (egocentric) point-of-view, while OadTR uses information from a third-person perspective. Thus, the action detection results from OadTR would not generalize to my Augmented Reality use-case.

5.3 Methodology

5.3.1 Proactive Augmented Reality Interaction Data of Cooperative Agents

Multimodal ego-centric vision-language benchmarks are provided in the HoloAssist dataset [319] that focus on Augmented Reality (AR)-based human-AI collaboration. This is the only open-source proactive dataset that captures Expert-User collaborative dynamics with AR devices. It records the visual observations of a Human User Agent interacting and collaborating with an Expert Instructor Agent, which can be an AI Agent, on physical reasoning tasks. 482 unique

Expert-User interaction video and dialogue sequences are present in the dataset, spanning 20 diverse task domains, including but not limited to:

- Cooking procedures like making coffee
- Fixing items like motorcycles
- Assembling/Disassembling furniture
- Assembling Devices like computers, GPUs, scanners
- Maintaining Electrical systems like circuit breakers
- Configuring Devices like printers, cameras, switches

The AR devices are worn by the user agent to capture first-person perspective videos while executing procedural tasks. The observations of the Expert Agent and guidance to the User Agents are simultaneously captured by the AR devices. The annotation schema of the dataset encompasses a variety of interaction types. Some examples of interactions done by the Expert Agent include:

1. Proactive Interactions

- High-level instructional guidance
- Follow-up instructions without any user input
- Feedback after intervening
- Steps to fix mistakes

2. Reactive interactions:

- Expert clarifications to user queries
- User-initiated dialogues

The HoloAssist data is obtained from 350 distinct expert-user interaction pairs leading to 45.5 hours of video recordings. It provides strong fundamentals to study the spatio-temporal dynamics of when and how Proactive Multimodal AI Agents should proactively engage in AR-assisted collaborative scenarios. This temporal aspect is important in developing AI agents which can guide humans in human-AI collaborative tasks when an AI Agent is yet to intervene, but should intervene with a goal to improve collaborative task execution or correct mistakes.

5.3.2 Generation from Multimodal VLM

Objects in video sequences are counted by leveraging the recent advancements in Multimodal Visual Language Models (VLMs). The proposed method extracts frames at a rate of 1 frame per second (FPS) by processing videos, enabling efficient temporal analysis while maintaining sufficient granularity for accurate object counting.

Frame Extraction and Processing Given an input video V of duration T seconds, a sequence of frames $\{f_1, f_2, \dots, f_T\}$ is extracted at 1 FPS. This sampling rate balances computational efficiency with temporal resolution, thereby ensuring that significant object state changes are captured while minimizing redundant processing.

Multimodal Generation PaliGemma-3b-mix-448, from the PaliGemma [33] family of lightweight multimodal VLMs created by Google, is used to process each frame independently. PaliGemma’s ability to quickly leverage its visual and textual understanding capabilities makes

it suitable for an AR / VR setting where there may not be much device compute available and a fast response is needed. For each frame f_t a prompt is constructed:

$$P_{obj} = \text{“The number of objects in this image is ”} \quad (5.1)$$

This prompt elicits a numerical response from the model, avoiding unwanted conversational output. This prompt was chosen after thorough experimentation with PaliGemma’s object detection capabilities. The model processes each frame f_t with prompt P_{obj} to generate a count estimate:

$$C_t = \text{PaliGemma}(f_t, P_{obj}) \quad (5.2)$$

where C_t represents the predicted object count at time t . The change in object count between frames is heavily skewed towards zero as seen in Figure 5.2.

Implementation Details The configuration used has 448×448 pixel resolution. After generating the model responses, minimal post-processing is performed to extract numerical values.

5.3.3 Alignment with Changing Object Count

The alignment signal that takes the form of a change in object count for the scene observed by the AI Assistant is motivated by an intuition of how humans operate when listening to instructions. A user agent will not be moving objects around when being guided on how to do a computer assembly while they process the instructions. Rather, they will be listening so they know what to do next. Building upon this understanding, the AI Assistant estimates when a proactive AI assistant should intervene by simply monitoring the change in object count from

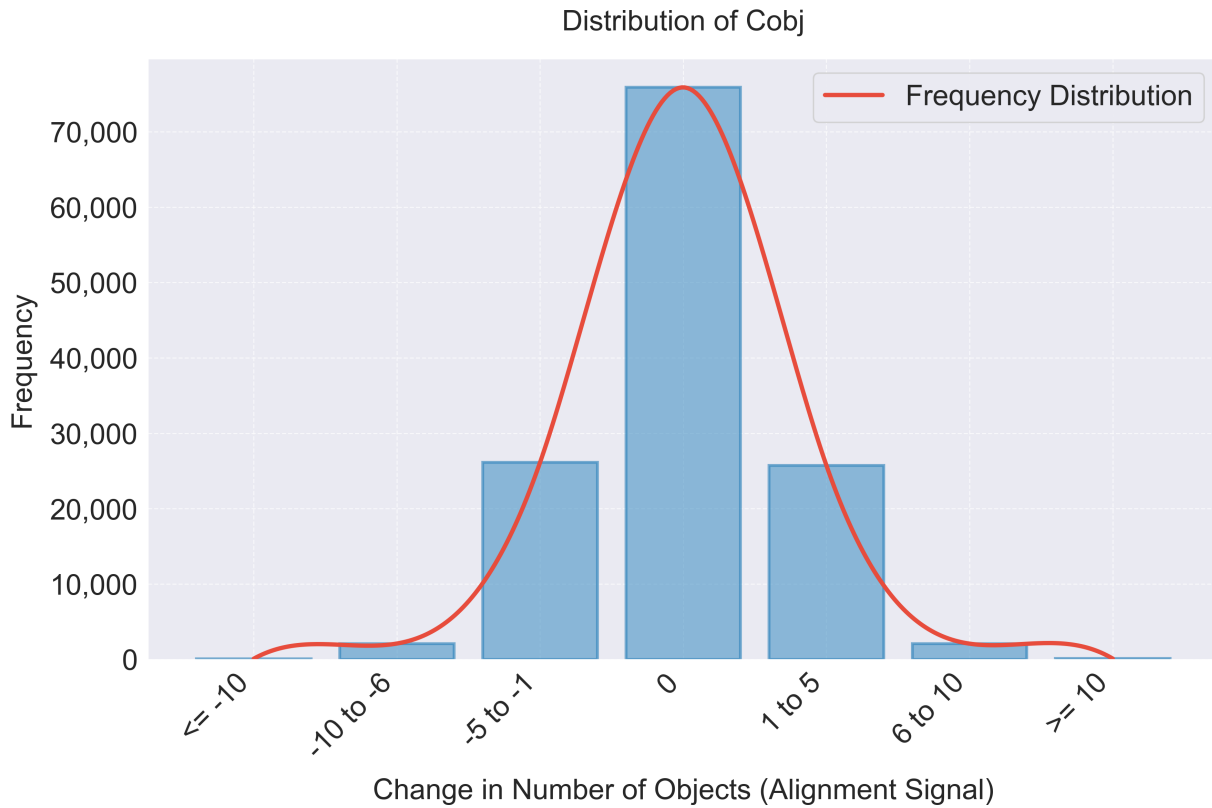
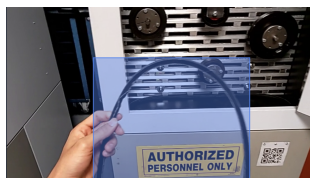


Figure 5.2: Distribution of Alignment Signal

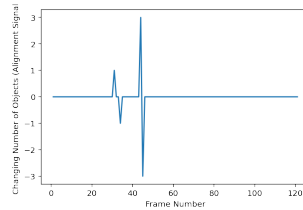
second to second. Figure 5.3 shows an example of frames where the object count changes.



(a) Second 36



(b) Second 37



(c) Alignment Signal over entire video.

Figure 5.3: A plot of the alignment signal over time for a video, accompanied by two consecutive video frames where a change in object count occurs. YETI detects a proactive intervention to help the user replace a mechanical belt on Frame 37 after observing the alignment signal from Frame 36. [17]

5.3.4 Spatio-temporal Signal Generation

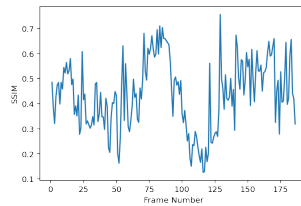
The frame-wise counting results are aggregated to create a temporal signal $\{\Delta C_1, \Delta C_2, \dots, \Delta C_{T-1}\}$ where $\Delta C_i := C_{i+1} - C_i$. This signal captures how the number of objects change throughout the video sequence.



(a) SSIM Second 98



(b) SSIM Second 99



(c) SSIM Signal over entire video.

Figure 5.4: SSIM signal is filtering proactive interventions by YETI Agent in an image frame (time instance) of a video observing a procedural task on how to assemble a RAM computer. Proactive intervention happens at the 98th second in Frame 98

5.3.5 Structural Similarity

The Structural Similarity Index Measure (SSIM) [322] is used to identify meaningful frames for proactive intervention by analyzing temporal coherence between consecutive frames. This approach helps filter out redundant frames where the scene remains largely static, such as Figure 5.4, thus focusing intervention detection on moments of significant change.

Given two consecutive frames f_i and f_{i+1} , their SSIM value is:

$$\text{SSIM}(f_i, f_{i+1}) = \frac{(2\mu_i\mu_{i+1} + c_1)(2\sigma_{i,i+1} + c_2)}{(\mu_i^2 + \mu_{i+1}^2 + c_1)(\sigma_i^2 + \sigma_{i+1}^2 + c_2)} \quad (5.3)$$

where μ_i, μ_{i+1} denote the mean intensity (brightness) of frames f_i and f_{i+1} , $\sigma_i^2, \sigma_{i+1}^2$ represent their respective variances, $\sigma_{i,i+1}$ is the covariance between the frames, $c_1 = (0.01L)^2$ and $c_2 = (0.03L)^2$ are stability constants, and $L = 255$ is the dynamic range of pixel values in grayscale.

The SSIM metric yields values in $[0, 1]$, where higher values indicate greater structural similarity between successive frames. This property identifies and filters out frames with SSIM values above a threshold τ , removing redundant temporal information. Interventions are triggered only during meaningful scene changes where a human user acts, reducing unnecessary proactive interventions while maintaining responsiveness to significant environmental variations. Similar to the temporal signal for changing object count, these SSIM values are consolidated for each frame (up to second T) into a set $\{s_1, s_2, \dots, s_{T-1}\}$.

5.3.6 Proactive Interactions and Interventions

Reactivity refers to the AI Agent's response to a user cue in the context of AI Agents. On the other hand, proactivity involves behaviors initiated by the AI Agent without user prompts. Proactive activities can be broadly categorized into proactive interaction and proactive intervention. It is essential to understand the subtle but important differences between these two categories in order to leverage the capabilities of a proactive agent. An AI Agent is considered to be *proactively interacting* with the user if it initiates any engagement without user cues. Conversely, an AI Agent is *proactively intervening* when it takes concrete steps to alter the user's behavior. By

this definition, all proactive interventions are forms of proactive interactions, but not all proactive interactions qualify as interventions.

HoloAssist includes eight distinct categories of proactive behavior, of which three are classified as interventions, as detailed in Table 5.2.

Conversation Type	Interaction	Intervention	Example
Follow-up Instruction	✓	✓	"Put the battery back."
Confirming Previous Action	✓	✓	"Perfect."
Correcting Wrong Action	✓	✓	"Nope, not that one."
Describe High-Level Instruction	✓	×	"You're going to validate, so we're going to move the shift lever through all of the settings."
Opening Remarks	✓	×	"Now for this task, we are removing the graphic cards from the PC slot."
Closing Remarks	✓	×	"You're all done. You can exit now."
Adjusting Video	✓	×	"Just keep your eyes on your hands."
Other	✓	×	"You can ground yourself again, it's not necessary."

Table 5.2: Examples of Proactive Interactions and Proactive Interventions in HoloAssist. [17]

5.3.7 YETI Proactive Agent Intervention Algorithm

The YETI algorithm incorporates several hyperparameters that determine when a Multimodal AI Agent should autonomously intervene proactively without any question or clarifications asked by the User Agent.

Algorithm 2 YETI Proactive Intervention Detection

Require: SSIM-filtered Frame sequence F with alignment scores $\{\Delta C_i\}$

Ensure: Set of intervention frames \mathcal{I}

```
1: Initialize empty sets:  $\mathcal{I}$ ,  $E_{obj}$  where  $E_{obj}$  stores alignment scores per episode of frames
2: Set episode interval  $k$ , conversation interval  $m$ , local extrema range  $r$ 
3: Initialize episode count  $n = 0$ , frame count  $t = 0$ 
4: for each frame  $f_i \in F$  do
5:   if  $n > 0$  and  $f_i \notin$  conversation interval then
6:      $t \leftarrow t + 1$ 
7:     Add  $\Delta C_i$  to  $E_{obj}$ 
8:     if  $c_i \in$  local extrema range then
9:        $\mathcal{I} \leftarrow \mathcal{I} \cup \{f_i\}$ 
10:    end if
11:    if  $t = k$  then
12:      Update conversation interval
13:      Reset episode metrics
14:       $n \leftarrow n + 1$ ,  $t \leftarrow 0$ 
15:    end if
16:  else if  $n = 0$  then
17:     $t \leftarrow t + 1$ 
18:    Add  $\Delta C_i$  to  $E_{obj}$ 
19:    if  $t = k$  then
20:       $\Delta C_{min} \leftarrow \min(E_{obj})$ 
21:       $\Delta C_{max} \leftarrow \max(E_{obj})$ 
22:      Define local extrema range with tolerance  $r$ .
23:       $\mathcal{I} \leftarrow \mathcal{I} \cup \{f_{current}\}$ 
24:      Update conversation interval
25:      Reset episode metrics
26:       $n \leftarrow n + 1$ ,  $t \leftarrow 0$ 
27:    end if
28:  end if
29: end for
    return  $\mathcal{I}$ 
```

- **SSIM threshold (τ):** This scene understanding parameter sets a filtering threshold to remove frames where user actions do not change or have not happened. SSIM value shows how close two consecutive image frames are. If a frame and its next frame have an SSIM of $\geq \tau$, the frame will not be considered for an autonomous intervention.
- **Conversation Interval (m):** This parameter constrains a minimum temporal gap between

consecutive interventions, reducing the frequency of intervention by AI. It defines the duration that must pass after an intervention before another can happen.

- **Local Extrema Range (r):** This parameter identifies the sensitivity of the YETI algorithm to changes in object counts along frames. It defines the range within which a change in object count must fall or rise to be significant.
- **Episode Interval (k):** This parameter limits the length of an "episode" to restrict the maximum rate of interventions. An episode is a consecutive sequence of frames where only one intervention is permitted, thus preventing excessive and disruptive interventions.

Algorithm 2 describes how the YETI agent detects proactive interventions. The input is a series of video frames from the perspective of the human user (pre-processed to filter out redundant frames with a high similarity) and the output is the instances where a proactive intervention may be needed.

5.4 Experiments

I conduct experiments with a wide variety of settings to obtain the results of my YETI algorithm. This allows understanding of how each configuration of the algorithm and the value of each hyperparameter contributes to the evaluation metrics of the YETI method compared to HoloAssist's approaches, the baseline for AI Agents proactively intervening with an user (student) task. An example of a proactive intervention being detected can be seen in Figure 5.5.

5.4.1 Experimental Settings

I carefully select hyperparameters to balance the trade-off between timely interventions and avoiding excessive interruptions. The key parameters are summarized in Table 5.3. A comprehensive analysis of hyperparameter sensitivity is provided in the detailed analysis of the results.

Parameter	Value
SSIM threshold (τ)	0.9
Conversation interval (m)	1
Extrema range (r)	± 1
Minimum history (k)	5

Table 5.3: Hyperparameters used in YETI experiments.

Settings used for different hyperparameters can be found in Table 5.3. τ controls frame similarity filtering, m sets minimum gap between interventions, r defines the range for local minima detection, and k specifies required history length before intervention.

I have evaluated two variants of the YETI algorithm:

- **Global YETI:** Uses the first detected local extrema as a fixed threshold throughout the video frame sequence
- **Local YETI:** Continuously updates the extrema threshold based on its recent history

This is useful to analyze the impact of adaptive versus fixed intervention thresholds on YETI’s performance. The full results of evaluating performance in this way can be seen in Tables 5.4 and 5.5.

For the comparative analysis between the proposed YETI methods and HoloAssist, it is important to note that evaluation of proactive intervention in real-time is limited to videos where the user and the expert agent both initiate conversations. This stipulation more closely aligns with

the real-world assistive AI Agent scenarios, where the user may ask for help in completing a task. This turns out to be 482 videos out of the total HoloAssist dataset. Results from the HoloAssist paper are also aggregated for each intervention class into one through averaging in order to have a head-to-head comparison with regards to predicting any kind of intervention. Detailed results additionally provide insights into the efficiency of the proactive AI Agents where the user agent does not communicate with any dialogue. However, the expert agent proactively interacts and intervenes with the user based on the user’s visual observations.

	Method	Overall			Confirm Action			Correct Mistake			Follow Up		
		Prec.	Rec.	F-score	Prec.	Rec.	F-score	Prec.	Rec.	F-score	Prec.	Rec.	F-score
HoloAssist	(RGB)	13.93	33.33	19.65	0.00	0.00	0.00	0.00	0.00	0.00	41.79	100.00	58.95
	(R+H)	24.89	33.64	28.61	32.14	4.50	7.89	0.00	0.00	0.00	42.52	96.43	59.02
	(R+E)	25.55	33.73	29.08	33.90	10.36	15.87	0.00	0.00	0.00	42.76	90.83	58.15
	(R+H+E)	48.31	37.59	42.28	39.11	40.93	40.00	61.11	9.91	17.05	44.70	61.93	51.92
	(RGB[Pt])	37.54	37.74	37.64	42.31	27.50	33.33	27.33	36.61	31.30	42.97	49.11	45.84
YETI (Ours)	Global	41.86	88.31	56.17	22.54	91.51	36.17	11.68	90.29	20.69	27.55	89.69	42.15
	Local	46.88	60.38	52.77	26.55	68.62	38.29	14.71	68.02	24.18	30.07	62.05	40.51

Table 5.4: Performance of YETI compared to HoloAssist in detecting proactive interventions.

5.4.2 Metrics

Evaluations demonstrate the YETI algorithm’s effectiveness in detecting appropriate moments for proactive interaction and intervention in real-time using metrics, also used in classification:

	Method	Overall			
		Acc.	Prec.	Rec.	F1
YETI (Ours)	Global	86.97	52.23	87.04	65.28
	Local	93.76	64.51	59.73	62.02

Table 5.5: Evaluation of YETI on the proactive interaction detection setting which encompasses the proactive intervention benchmark

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (5.4)$$

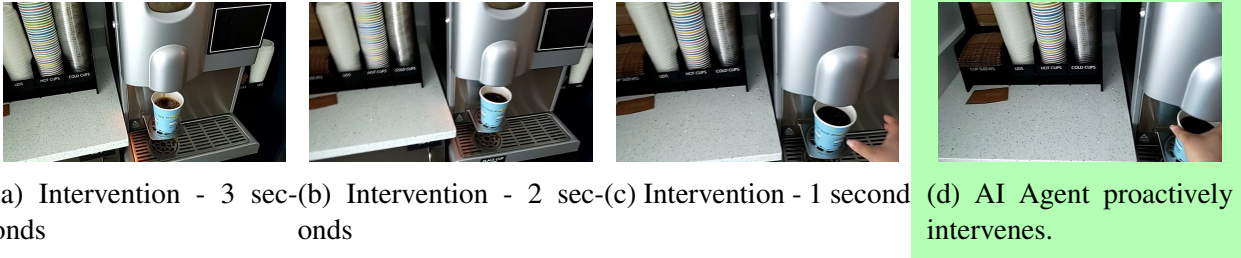
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5.5)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (5.6)$$

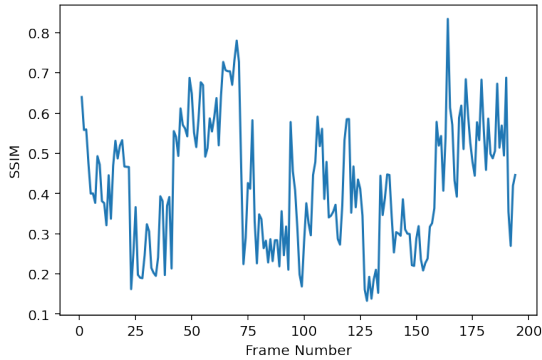
$$\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.7)$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. These metrics provide a comprehensive assessment of YETI’s intervention capabilities, measuring both its ability to intervene at appropriate moments and its capacity to avoid unnecessary interruptions.

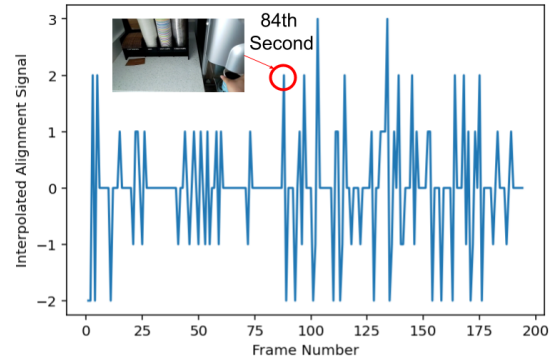
As there is no empiric measurement of when the “best” precise moment to intervene is, a window of five seconds is considered around HoloAssist labeled ground-truth proactive intervention starting time-stamps in order to assess whether a detected intervention frame is a true positive or a false positive. This is consistent with the method used to evaluate the HoloAssist baseline model [319], which also determines true positives and false positives by measuring temporal proximity to the nearest labeled intervention and seeing if it is within a window of tolerance.



(a) Intervention - 3 seconds (b) Intervention - 2 seconds (c) Intervention - 1 second (d) AI Agent proactively intervenes.



(e) SSIM between consecutive frames.



(f) Changing Objects Alignment Signal

Figure 5.5: Proactive Intervention Detection for Coffee Making Task. [17]

5.4.3 Results

It can be seen in Tables 5.5 and 2 that YETI achieves impressive performance to detect proactive interventions and detections on-the-fly. This performance is much better than HoloAssist’s baselines which uses more complicated methods. Furthermore, as PaliGemma’s scale is on the order of three billion parameters, it is suitable to be used in an Augmented Reality or an edge device. The small data size of SSIM and Object Count Alignment relative to other features, as seen in Table 5.1, allows AI Agents on-device to use the information effectively saving memory and power while having a small response time.

5.5 Detailed Analysis of Results

This section provides a more in-depth analysis of the results presented in Tables 5.1, 5.4, and 5.5 with a special focus on Table 5.4 comparing the performance of YETI to the HoloAssist baseline in detecting proactive interventions.

Overall Performance:

YETI consistently outperforms HoloAssist [319] overall, demonstrating a significant improvement in accurately detecting proactive interventions in most scenarios. This superior performance is observed in both the Global and Local variants of YETI. Notably, YETI achieves substantially higher recall in all categories, indicating its effectiveness in identifying a greater proportion of actual proactive interventions. This improvement is crucial for ensuring that the AI agent can effectively assist users by recognizing and responding to their needs in a timely manner.

Specific Intervention Types:

- **Overall:** When taking into account all intervention types, both Local and Global variants of YETI substantially outperform the HoloAssist baseline on recall and F-measure, while surpassing precision in every modality of HoloAssist except R+H+E (RGB video + Hand pose + Eye gaze), where it still remains competitive.
- **Confirm Action:** In only considering interventions where the AI Agent should step in to confirm the action the user is taking, YETI continues to achieve substantially higher recall than HoloAssist. The F-measure, which seeks to balance out precision and recall, is also higher than every combination of modalities used for HoloAssist besides R+H+E.

- **Correct Mistake:** While both YETI and HoloAssist struggle to detect when to intervene to correct a mistake by the user, many modalities for HoloAssist fail to detect them entirely, and for the ones that can detect some of these interventions, the recall is very poor. YETI, on the other hand, has extremely high recall but comparatively poor precision. This contributes to YETI having a higher F-measure than all HoloAssist modalities besides HoloAssist RGB[Pt] (model pretrained on ImageNet).
- **Follow Up:** Following up with the user after the user performs an action is where YETI struggles the most compared to the HoloAssist baselines, but it still remains competitive, outperforming the R+H+E and RGB[Pt] baselines on recall. Ironically, even though YETI and R+H+E and RGB[Pt] performed better on every other category, they had the worst results in following up. This indicates that more sophisticated features may complicate the process of detecting Follow-up interventions.

Further Comparison with HoloAssist:

The performance gains of YETI over HoloAssist can be attributed to several factors:

- **Global / Local Context:** YETI's ability to leverage its history contributes to its improved accuracy in detecting proactive interventions. By considering both the broader context of the interaction, as in the case of Global YETI, and the specific details of the current situation, as in the case of local YETI, YETI can better understand the user's intentions and needs.
- **Advanced Feature Representation:** YETI likely employs more streamlined feature representations compared to HoloAssist, allowing it to capture more nuanced aspects of the interaction and make more informed decisions about potential interventions.

- **Data Efficiency:** The smaller data size of SSIM and Object Count Alignment features, as highlighted in Table 5.1, enables YETI to process information and respond to user actions with minimal latency. This efficiency is crucial for real-time applications in augmented reality environments.

YETI's superior performance in detecting proactive interventions, coupled with its efficiency and suitability for deployment on augmented reality devices, makes it a promising approach for enhancing human-computer interaction in various domains. Its ability to accurately detect when to proactively intervene has the potential to significantly improve user experiences and facilitate more effective collaboration between humans and AI agents.

5.6 Comparative Analysis of YETI

5.6.1 Comparison with other Classifier Models

A Random Forest Classifier, a Decision Tree, and a Multi-Layer Perceptron (MLP) were trained in order to assess the efficacy of my YETI algorithm (Algorithm 2). The train and the test data statistics for the Random Forest Classifier can be seen in Table 5.6. Due to the imbalance in labels, it was more statistically safe for the models to assume that every single frame was a negative example of a proactive interaction or a proactive intervention, resulting in accuracy being high but precision, recall, and F-measure being zero. This justifies the use of my algorithm to detect when to intervene rather than relying solely on a model such as these ones, which are sensitive to skewed data distributions.

As seen in Table 5.6, Random Forest (RF), Decision Tree (DT) and Multi-Layer Percep-

	Interactions	Interventions	Conf. Action	Corr. Mistake	Follow Up	Total
Train	7285 (6.77%)	4562 (4.24%)	2222 (2.07%)	526 (0.49%)	1814 (1.37 %)	107592 (81.43%)
Test	1583 (6.45%)	1005 (4.1%)	474 (1.93%)	142 (0.58%)	389 (1.59%)	24537 (18.57%)
Total	8868 (6.72%)	5567 (4.21%)	2696 (2.04%)	668 (0.51%)	2203 (1.67%)	132129

Table 5.6: Training and Test splits of Image Frames with Distributions of Proactive Interactions, Proactive Interventions and 3 Proactive Intervention types

	Interactions	Interventions	Confirm Action	Correct Mistake	Follow Up
Rand. Forest					
Decision Tree	93.55	95.9	98.41	99.42	98.07
MLP					
Global YETI (Ours)	86.97	84.85	80.75	79.97	82.36
Local YETI (Ours)	93.76	93.36	93.05	93.32	93.31

Table 5.7: Accuracy of Classification Models in Proactivity Prediction.

tron (MLP) Classifiers have high accuracies as they can predict when the AI Agent should not be proactively interacting or intervening. The same trend is observed in the 3 different proactive intervention type predictions like confirming actions, correcting mistakes, following up instructions. However, RF, DT and MLP classifiers cannot classify any true positive (actual proactive instances) due to the extremely sparse and skewed distribution of proactivity labels in Table 5.7, thus has 0 precision and recall. These classifiers cannot detect proactivity on-the-fly in real time. My YETI Proactive Agent Detection Algorithm has good precision and high recall model architectures in detecting proactivity on-the-fly as shown in Table 5.4.

5.6.2 Comparison with Implicit Expert-User Agent Setting

In Table 5.4, only results using data from HoloAssist examples with both student-led and instructor-led conversations were used. In Tables 5.8 and 5.9 are the results for the 1191 HoloAs-

sist examples where there was not a student-led conversation, instead relying on implicit student-instructor interactions where, for example, the instructor tells the student to do something and the student complies without further comment.

	Method	Overall			Confirm Action			Correct Mistake			Follow Up		
		Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.
HoloAssist	(RGB)	13.93	33.33	19.65	0.00	0.00	0.00	0.00	0.00	0.00	41.79	100.00	58.95
	(R+H)	24.89	33.64	28.61	32.14	4.50	7.89	0.00	0.00	0.00	42.52	96.43	59.02
	(R+E)	25.55	33.73	29.08	33.90	10.36	15.87	0.00	0.00	0.00	42.76	90.83	58.15
	(R+H+E)	48.31	37.59	42.28	39.11	40.93	40.00	61.11	9.91	17.05	44.70	61.93	51.92
	(RGB[Pt])	37.54	37.74	37.64	42.31	27.50	33.33	27.33	36.61	31.30	42.97	49.11	45.84
YETI (Ours)	Global	41.86	88.31	56.17	22.54	91.51	36.17	11.68	90.29	20.69	27.55	89.69	42.15
	Local	46.88	60.38	52.77	26.55	68.62	38.29	14.71	68.02	24.18	30.07	62.05	40.51
Implicit	Global	28.96	92.59	44.11	15.68	94.76	26.90	11.28	93.58	20.12	20.47	92.56	33.53
	Local	34.55	71.71	46.63	18.40	77.71	29.76	14.28	74.21	23.95	24.51	72.48	36.64

Table 5.8: Evaluation of YETI compared to HoloAssist and Implicit Interventions

5.7 Ablation Studies

5.7.1 Agent Conversation Interval

The Agent Conversation Interval is a parameter for it will probably take a user to respond to an intervention by the AI Agent. Table 5.3 uses a value of one, indicating the user will take about one second to act on the advice of the proactive AI Agent. Here alternative intervals are

	Method	Overall			
		Acc.	Prec.	Rec.	F1
YETI (Ours)	Global	86.97	52.23	87.04	65.28
	Local	93.76	64.51	59.73	62.02
Implicit	Global	81.37	35.68	91.9	51.42
	Local	92.62	45.67	71.28	55.67

Table 5.9: Evaluation of YETI compared to HoloAssist and Implicit Interactions

explored where a user might respond to the intervention within up to five seconds rather than one second.

Table 5.10 shows precision, recall, and F-measure for the detection of each kind of proactive intervention, with varying values for the conversation interval. As the conversation interval is increased, the F-measure decreases, showing that overall, a conversation interval of one is the best choice. The same logic follows for detection proactive interactions, shown in Table 5.11, where it is observed that F-measure decreases as the conversation interval increases.

Method	Conv. Interval	Overall			Confirm Action			Correct Mistake			Follow Up		
		Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.
Global YETI	1	41.86	88.31	56.17	22.54	91.51	36.17	11.68	90.29	20.69	27.55	89.69	42.15
	2	41.06	84.09	55.18	22.52	88.33	35.89	11.49	86.52	20.29	27.44	85.87	41.59
	3	41.13	80.49	54.44	22.41	85.42	35.50	11.50	83.28	20.22	27.31	82.53	41.04
	4	41.08	77.09	53.61	22.53	82.81	35.43	11.54	80.46	20.19	27.33	79.34	40.65
	5	40.95	73.91	52.70	22.31	80.03	34.89	11.59	77.92	20.19	26.96	76.22	39.83
Local YETI	1	46.88	60.38	52.77	26.55	68.62	38.29	14.71	68.02	24.18	30.07	62.05	40.51
	2	47.01	56.40	51.27	26.54	64.97	37.69	13.92	63.37	22.82	30.39	58.54	40.01
	3	47.15	56.39	51.36	25.93	64.46	36.99	14.35	63.52	23.41	30.75	59.26	40.49
	4	46.43	53.66	49.78	25.78	62.39	36.49	13.32	58.98	21.73	30.41	56.45	39.53
	5	45.81	51.01	48.27	26.05	60.58	36.44	13.37	56.71	21.64	29.47	52.99	37.88

Table 5.10: Evaluation of YETI for Agent Conversation Intervals for proactive interventions

5.7.2 Extrema Range

The extrema range refers to how close the alignment score needs to be to a minima or maxima (whether local or global) in order for a proactive intervention to be detected. A larger extrema range makes YETI more sensitive, resulting in more proactive detections, while a narrower extrema range leads to less proactive detections. In Tables 5.12 and 5.13 an extrema range of zero results in a much lower recall, while an extrema range of two results in a lower precision. Since overall, YETI struggles more with precision than recall, the chosen extrema range is one,

Method	Conv. Interval	Overall			
		Acc.	Prec.	Rec.	F1
Global YETI	1	86.97	52.23	87.04	65.28
	2	90.29	52.30	82.51	64.02
	3	92.01	52.42	78.63	62.90
	4	93.11	52.69	75.20	61.97
	5	92.65	51.15	76.22	61.22
Local YETI	1	93.76	64.51	59.73	62.02
	2	94.25	63.79	55.47	59.34
	3	94.59	63.43	55.12	58.98
	4	94.77	62.23	52.3	56.84
	5	94.98	61.61	49.79	55.08

Table 5.11: Comparison over different Agent Conversation interval sizes for proactive interactions.

which maximizes precision.

Method	Extrema Ranges	Overall			Confirm Action			Correct Mistake			Follow Up		
		Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.
Global YETI	0	41.34	70.61	52.15	22.52	76.96	34.85	11.09	73.87	19.28	27.13	73.92	39.69
	± 1	41.86	88.31	56.17	22.54	91.51	36.17	11.68	90.29	20.69	27.55	89.69	42.15
	± 2	41.05	91.95	56.75	22.19	94.09	35.91	11.48	93.08	20.44	27.43	93.04	42.37
Local YETI	0	43.42	26.02	32.54	24.83	33.16	28.39	11.12	28.28	15.96	27.35	27.17	27.26
	± 1	46.88	60.38	52.77	26.55	68.62	38.29	14.71	68.02	24.18	30.07	62.05	40.51
	± 2	46.74	77.23	58.24	25.98	83.11	39.59	13.34	79.83	28.86	30.63	79.22	44.18

Table 5.12: Evaluation of YETI for Extrema Ranges for proactive interventions

Method	Extrema Ranges	Overall			
		Acc.	Prec.	Rec.	F-meas.
Global YETI	0	92.14	53.94	68.52	60.36
	± 1	86.97	52.23	87.04	65.28
	± 2	84.82	51.76	91.09	66.02
Local YETI	0	94.40	68.19	28.17	39.87
	± 1	93.76	64.51	59.73	62.02
	± 2	91.82	61.39	76.09	67.95

Table 5.13: Comparison over different Extrema ranges.

5.7.3 SSIM Thresholding

When certain frames are filtered based on their SSIM score relative to the previous frame, the meaning of what it means to have structural similarity is made more abstract. This abstraction is a form of reasoning.

Method	SSIM	Overall			Confirm Action			Correct Mistake			Follow Up		
		Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.
Global YETI	0.5	43.20	71.83	53.95	23.86	77.14	36.45	11.40	71.87	19.68	30.01	76.74	43.14
	0.6	42.55	79.60	55.46	23.07	83.90	36.19	11.70	81.33	20.46	29.53	83.02	43.56
	0.7	41.71	85.49	56.06	22.56	88.93	35.99	11.88	87.77	20.93	28.81	87.69	43.36
	0.8	41.29	87.38	56.08	22.44	90.64	35.98	11.87	89.37	20.96	27.99	89.05	40.59
	0.9	41.86	88.31	56.17	22.54	91.51	36.17	11.68	90.29	20.69	27.55	89.69	42.15
Local YETI	0.5	47.35	43.05	45.09	25.88	50.31	34.18	14.23	45.22	21.65	32.66	48.00	38.87
	0.6	45.55	49.19	47.30	24.60	56.28	34.24	14.00	54.62	22.29	30.84	53.22	39.05
	0.7	45.93	55.86	50.41	25.09	63.09	35.91	14.50	63.14	23.59	30.44	58.83	40.12
	0.8	47.09	59.52	52.58	26.50	67.61	38.08	14.98	67.39	24.51	30.45	61.47	40.73
	0.9	46.88	60.38	52.77	26.55	68.62	38.29	14.71	68.02	24.18	30.07	62.05	40.51

Table 5.14: Evaluation of YETI for SSIM Thresholding for proactive interventions

Method	SSIM	Overall			
		Acc.	Prec.	Rec.	F-meas.
Global YETI	0.5	91.49	55.38	68.88	61.39
	0.6	90.20	54.14	77.36	63.69
	0.7	88.62	53.06	83.92	65.01
	0.8	87.55	52.39	85.99	65.11
	0.9	86.97	52.23	87.04	65.28
Local YETI	0.5	94.13	64.25	41.12	50.14
	0.6	93.98	62.75	48.05	54.42
	0.7	93.88	63.46	55.14	59.01
	0.8	93.81	64.42	58.65	61.40
	0.9	93.76	64.51	59.73	62.02

Table 5.15: Comparison over different SSIM Thresholding levels for proactive interactions.

5.7.4 Episode Length

Varying the initial history (i.e. Episode Length) needed in order to make the first proactive intervention/interaction did not influence the results from those in Tables 5.4 and 5.5. This could be due to the fact that the Episode Length is only relevant for the first few frames of the video, and thus has an extremely narrow window to influence the ultimate output.

5.8 More Use Cases for Proactive Intervention Detection

The HoloAssist paper [319] presents many additional use cases for an AI Assistant that can proactively intervene when the user is trying to accomplish a task. An exhaustive list of all the uses included in the dataset is as follows:

- Assembly tasks
 - Assembling a computer
 - Assembling a laser scanner
 - Assembling a nightstand
 - Assembling a stool
 - Assembling a tray table
 - Assembling a utility cart
 - Disassembling a nightstand
 - Disassembling a tray table
 - Disassembling a utility cart

- Maintenance and Repair
 - Changing a mechanical belt
 - Changing a circuit breaker
 - Fixing a motorcycle

- Consumer Electronics
 - Making coffee with an espresso machine
 - Making coffee with an espresso machine
 - Setting up a big printer
 - Setting up a camera
 - Setting up a GoPro
 - Setting up a small printer
 - Setting up a Nintendo Switch

Assistive AI Agents that can proactively intervene are particularly useful for tasks such as these, where they can help with:

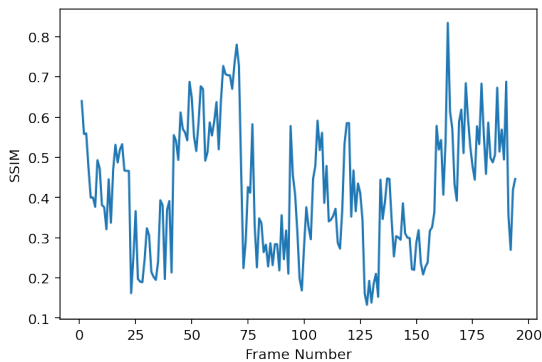
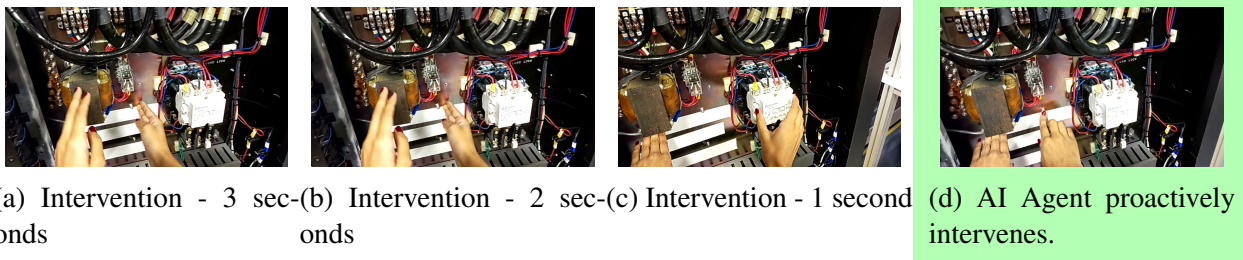
1. **Complexity Management:** Furniture assembly is a good example of a task that seems straightforward at first but can quickly get out of hand, especially if a mistake made early on is only noticed by the user much later on in the process. A proactive AI Agent intervention could help catch these mistakes early on before they become a larger problem.

2. **Safety Considerations:** Tasks such as changing a mechanical belt or a circuit breaker can open the user up to serious harm if they are not careful. If the user interacts with a live wire

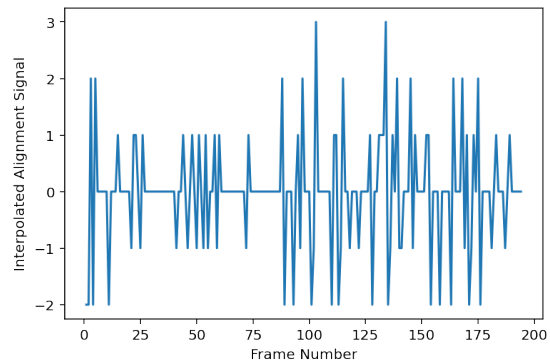
while the power is on they could be seriously injured. An AI assistant could potentially proactively intervene before the user makes this mistake.

3. **Expertise Gap:** If a user is new to completing a task that benefits from learned experience, the proactive AI Agent could take on the role of an instructor guiding a student. This would be especially useful for fixing a motorcycle, which is often done by expert mechanics.

The diversity of these use cases demonstrates that proactive AI assistance has broad applicability in scenarios where expert oversight would traditionally be beneficial. This is particularly valuable in contexts where users might not possess sufficient domain knowledge to self-identify potential issues or formulate appropriate queries for assistance.



(e) SSIM between consecutive frames.



(f) Changing Objects Alignment Signal

Figure 5.6: Intervention Detection for Changing Electric Circuit Task.

A Proactive AI Agent can help in many industrial or home maintenance tasks like how to change an electrical task in Figure 5.6. The AI Agent helps to guide the user on how to change

the circuit breaker and proactively intervenes without any questions from the user when the AI Agent observes that the user may touch the electric circuit which will be a safety worry. The Extrema of the changing object count for the corresponding frame is helpful to determine when the AI Agent should intervene proactively, post filtering by the SSIM signal. Filtering helps to obtain an abstract scene understanding capability to the Proactive AI Agent. The expectation is that the frames where the AI Agent should intervene are dissimilar and can be observed by SSIM while pruning the edge cases by filtering.

5.9 Conclusion and Future Work

In this Chapter, the significant gap in proactive AI assistance for everyday real-world tasks has been tackled by introducing the Yet-to-Intervene (YETI) agentic algorithm. Traditional AI agents operate reactively, responding only to explicit user prompts. This limits their effectiveness in dynamic and context-sensitive environments where anticipatory intervention can substantially enhance user experience and task efficiency. The YETI algorithm addresses this limitation by enabling AI agents to proactively identify and intervene in user actions without awaiting explicit cues.

YETI has been integrated with lightweight Vision-Language Models (VLMs) like PaliGemma in this approach, demonstrating remarkable improvements across key performance metrics, especially Recall, Accuracy, and F-Measure. YETI achieves these enhancements while using features that are up to 6,870 times more memory-efficient than those employed by state-of-the-art models such as HoloAssist. This drastic reduction in overhead not only makes YETI more accessible for deployment on resource-constrained devices but also paves the way for real-time,

scalable AI assistance in diverse augmented reality (AR) applications.

I have provided extensive ablation studies which reveal the critical features that influence the accurate detection of interventions. The most influential components that empower YETI to discern when proactive interactions or interventions are warranted are identified by systematically varying and analyzing different feature sets. This insight shows the importance of feature selection in the design of efficient and effective AI assistance algorithms.

Looking ahead, there are several promising avenues to carry on for future research. First, YETI’s intervention capabilities can be enhanced by incorporating richer sensory data, including hand pose, eye gaze, head orientation, Inertial Measurement Unit (IMU) readings, and depth information. Integrating these modalities is expected to provide a more comprehensive understanding of the user’s context and intentions, thereby enabling more nuanced and timely interventions. I also plan to evaluate YETI’s performance across a broader spectrum of VLMs to assess its generalizability and identify optimal model architectures for proactive assistance. Additionally, fine-tuning VLMs on the HoloAssist dataset could further refine the agent’s ability to anticipate user needs and improve intervention accuracy.

Future implementations of YETI will explore adaptive learning mechanisms that allow the AI agent to refine its intervention strategies based on user feedback and evolving task dynamics. This adaptive approach is anticipated to enhance the personalization and effectiveness of AI assistance, making it more attuned to individual user preferences and behaviors. VLM capabilities can also be expanded, including using reasoning models to do semantic understanding, reasoning, and planning.

In summary, the YETI algorithm represents a significant advancement in the development of proactive AI assistants, offering enhanced performance with reduced computational demands.

YETI has the potential to transform human-AI collaboration in AR environments by enabling timely and context-aware interventions. Ongoing and future work seeks to build upon these foundations, further pushing the boundaries of what proactive AI agents can achieve in facilitating and automating complex real-world tasks.

Chapter 6: Model Predictive Control to Guide Self-Driving Cars

Control problems for autonomous AI agents, especially safety-critical applications such as autonomous vehicle control, require robust decision making frameworks to ensure safe navigation in such complex and dynamic environments. This necessitates approaches such as Agentic Model Predictive Control (MPC), which can anticipate future problems and plan for them accordingly. Recently research on Multimodal Vision Language Models (VLMs) have been helpful to improve semantic scene understanding based on the observation's context from extremely large amounts of information. These models vary in a wide range of sizes, trading off speed with performance as they scale further and further. Here I introduce a novel framework that integrates MPC with Multimodal VLMs [19] in order to enhance the ability of autonomous vehicles to navigate and respond to real-world scenarios. Leveraging the open-source Waymax library released by Waymo, along with Waymo Open Motion, Berkeley DeepDrive and NuScenes Datasets, this method uses Multimodal VLMs to detect and draw bounding boxes around important parts of the scene, such as pedestrians or other vehicles. These models are helpful for querying specific attributes of identified objects, such as telling if a vehicle is accelerating or decelerating, or by recognizing if a newly detected obstacle is on a collision course with the vehicle. By incorporating these and other semantic insights into an MPC framework, an autonomous vehicle can make more informed and more context aware decisions to mitigate the risk of a collision and

safely navigate its surroundings. This approach is evaluated in diverse simulated environments using VLMs of different scales, demonstrating improvements in safety and reliability metrics compared to traditional MPC methods. The integration of VLMs with MPC represents a significant advancement in autonomous decision making, and especially in dynamic and uncertain situations. This approach paves the way for future research in using Multimodal VLMs for more intelligent and adaptable autonomous agents.

6.1 Introduction

Autonomous vehicle control is a long-standing challenge in robotic reasoning and planning. Driving, for example, is feasible for most people given sufficient training and instruction, but is an extraordinarily complex task for a computer to accomplish [48]. There are countless data to take into account when deciding which action to take, and a severe departure from a planned course of action may be required at any moment. For this reason, reliable autonomous vehicles have remained elusive despite significant advances in Artificial Intelligence (AI) research.

One method used to implement planning in autonomous vehicles is Model Predictive Control (MPC). In MPC, the system has an explicit model of its own evolution, allowing it to anticipate future states and plan accordingly, adjusting as needed to any new information it receives. It works even in a complex multivariate environment, such as autonomous driving, where route tracking and vehicle movement must be taken into account [6]. MPC is a powerful and efficient framework used not only in autonomous vehicle control but in a wide variety of safety-critical applications such as chemical synthesis and air traffic control.

Vision Language Models (VLMs) have emerged as a popular way to create AI systems

capable of taking in different modalities of data similar to how a human would [114]. A VLM can ingest text, speech, images, video [20], and a host of other data to generate an output. Due to the vast amount of data used to train these models, VLMs are often used to give an AI a general world understanding, though they can also be fine-tuned for a particular task. The capabilities of a VLM scale considerably with the number of parameters, but an increase in size also comes with considerable latency, creating a tradeoff between performance and speed when it comes to model scale. For this reason, many VLMs often come packaged in a “family”, leaving it to the user to select which one best fits their use case.

In my work on Multimodal Agentic Model Predictive Control [19], I propose combining MPC and VLMs to create autonomous vehicle agents that have a semantic understanding of the world. The goal of this integrative approach is to create AI agents that will:

1. Have more context-aware decision making and improved safety in complex and dynamic situations
2. Be able address real-time decision making problems with powerful AI capabilities
3. Leverage knowledge from pre-training that can handle unforeseen circumstances better than traditional Autonomous Vehicle Control
4. Have superior interactions with humans by using language-based signals

This approach is not only possible but can be accomplished using open-source data such as the Waymo Open Motion Dataset [95] and the NuScenes Autonomous Driving dataset [47], as well as open-source simulators such as Waymax [122]. Multimodal Agentic Model Predictive Control gives fine-grained control which can be useful for the AI Agent adapting to mistakes and

navigating efficiently while training or finetuning self-driving vehicles.

6.2 Background

AI Agents are systems where an Artificial Intelligence is endowed with the ability to observe an environment, choose an action, and execute that action within the environment. The most important feature distinguishing AI Agents from traditional AI is that the AI Agent needs to be able to consider how its own behavior affects the world around it, rather than simply focusing on giving the "correct" output at any given moment. For this reason, Autonomous Vehicles, and particularly Self-Driving Cars, fit neatly into the category of AI Agents.

Autonomous Vehicle Development has been an active area of research since as far back as the 1980s [88]. While rule-based approaches were originally the norm for these systems, the advent of practical deep learning led to the quick adoption of advanced learning-based approaches that could capture patterns in data that could not be explicitly modeled by a set of human-written rules [267]. Self-driving car companies such as Waymo, Kodiak, and Wayve have released blog-posts with limited detail on using VLMs for Autonomous Vehicles, but they have not combined VLMs with classical control mechanisms like MPC which has been proposed in this Chapter.

MPC is a paradigm adopted both in rule-based approaches to artificial intelligence as well as learning-based approaches. Due to its ability to anticipate future scenarios and plan in dynamic environments, it is well suited for an environment as sophisticated as that of a driver operating a vehicle [313]. Another advantage of MPC is its interpretability, being that they:

1. Optimize a cost function that has a physically-grounded meaning
2. Make derivations entirely according to the constraints and objectives of the problem

3. Have predictions that can be readily analyzed

Recent research has also proposed merging MPC with other methods of control such as Stanley-based control [6].

VLMs have increasingly been used to take actions, with Vision-Language-Action (VLA) models emerging to address this use-case [160, 356]. Specifically, VLMs and controllers have been used in conjunction with one another before [35, 352]. In fact, VLMs have been used to serve as controllers in and of themselves [228, 355]. Previous work has looked at using MPC and VLMs in combination with one another for Self-Driving Cars [190]. I hypothesize a framework that can serve as a basis for other Autonomous Vehicle paradigms as well, including Autonomous Drones and Autonomous Aquatic Vehicles and Submersibles. With this proposed framework, it is aimed to further expand the horizon of what VLMs can be used for and improve the state-of-the-art for Autonomous Vehicle Control by introducing these new capabilities.

6.3 General Approach: VLM-MPCs for Autonomous Vehicles

6.3.1 Conceptual Framework

The integrative MPC=VLM framework for an enhanced Autonomous Vehicle Control is centered on the potential for improving performance by leveraging the semantic understanding of the world embedded in a VLM with MPC that can safely anticipate and plan for future challenges faced by the autonomous vehicle using information perceived by the VLM. At each timestep, the autonomous vehicle would have information input to its sensors. This input would be fed into the context of the VLM, along with a sufficient history of prior information. The system

would then query the VLM for information regarding the scene, such as whether there is a car in the scene or not, as well as changes in the scene, such as whether the car appears to be in the way of a pedestrian. A VLM of a size that the Autonomous Vehicle is capable of safely using given resource and time constraints should be used, as the vehicle must be able to quickly make decisions. A key advantage of this approach is that it gives humans a way to understand how the vehicle is planning for future obstacles, accomplishing the objective of a safe and trustworthy autonomous vehicle controller. An overview of the general framework is illustrated in Figure 6.1.

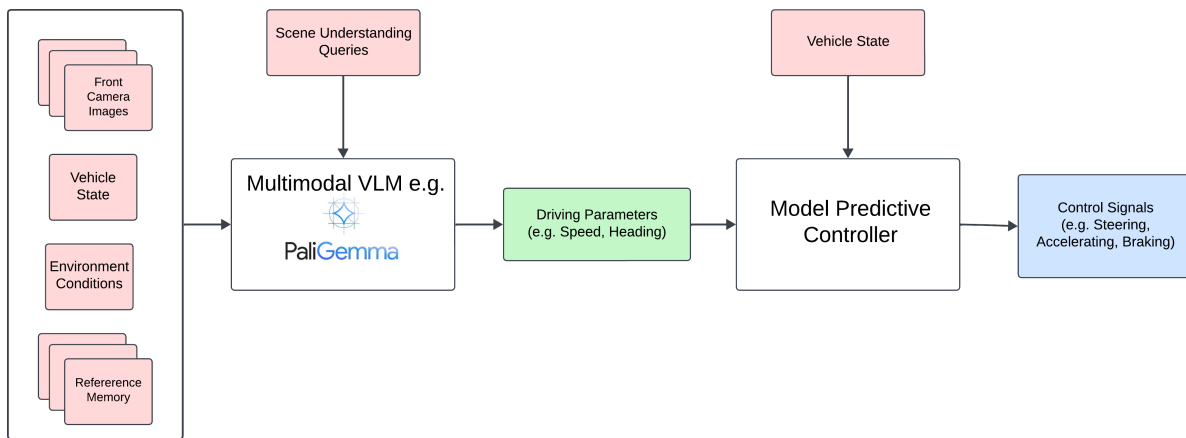


Figure 6.1: Overview of the decision making process for the VLM-MPC system for Autonomous Vehicle Control.

6.3.2 Scene Understanding using VLMs

While traditional computer vision techniques analyze a scene using statistical correlation, VLMs are capable of a higher-level understanding of contextual relationships [360]. VLMs have previously been used for scene understanding in the context of autonomous driving, even when given unusual visual data such as LIDAR [181] which is a common modality of autonomous vehicle data. These approaches have not been integrated with Autonomous Vehicle controllers,

but the potential for VLMs to enhance Autonomous Vehicles is clear. Relatively small VLMs, such as PaliGemma 3B [33] are capable of taking input frames from a video and detecting a particular kind of object, such as a car. They are also capable of segmenting the environment into its most important components, which makes them suitable for extracting a large amount of information from their output given a suitable query.



Figure 6.2: PaliGemma 3B can accurately identify and detect moving cars in a video.

6.3.3 Semantic Querying for Object Attributes

In addition to a general understanding of the world that can be provided through VLMs, there is additional information gained from making queries to the VLM about important objects detected in the video. If the VLM detects a car for example, it could be asked whether the car appears to be heading closer or further away, whether the vehicle is on a collision course with a

traffic cone or whether the car safely out of the way, along with many other pieces of information that could be extracted from the scene. This step leverages the semantic knowledge of the VLM in order to learn more about each object in the scene.

6.3.4 Integration of Semantic Insights into MPC

Once semantic insights on the scene observed by the VLM are obtained, it is important to find a way to integrate them into the MPC that makes full use of the information available while also ensuring that information is usable by the MPC. The output of the VLM can be formatted in terms of driving parameters such as the Speed and Heading of objects of note such as other vehicles. This information, combined with the known state of the Vehicle, can better inform the MPC and lead to better performance due to the additional information about the environment.

6.3.5 Real-time Processing and Decision Making

Due to the flexibility of the integrative VLM-MPC framework, VLMs of various scales and sizes can be applied to see which VLMs allow for the best balance of robust information and speed. Autonomous vehicles are absolutely required to make real-time decisions, as a life-or-death situation could come at any moment. With this in mind, larger models may be infeasible, while ultra-small models such as those designed for mobile phones may not fit into the right semantic capabilities required for the MPC to make good decisions. These different scales of VLM can be explored to get a model size that fits the requirements for both robust decision making and real-time decision making. Processing the information will also need to be done quickly in order for the VLM output to be realized in time in the first place, meaning the other

inputs to the VLM such as the vehicle state, environment conditions, and reference memory must also be represented in a compact format that can be processed quickly.

6.3.6 Safety-Centric Approach

As the main goal of integrating VLMs with a MPC is to provide control signals for an autonomous vehicle, safety must be of paramount importance. In order to reach safe control signals, additional safety constraints are imposed on the MPC. This can prevent the MPC from accelerating too much or from turning too sharply. This adds an additional layer of robustness to the controller, as output from the VLM still has the potential to be off due to hallucinations and other inaccuracies. There is a robust body of literature on imposing these sorts of constraints, but specifically engineering constraints for information sourced from the VLM can be another contribution of implementing this framework.

6.3.7 Adaptability and Generalization

With a VLM-backed MPC system, there is a significant ability for the autonomous vehicle to be able to adapt to unseen scenarios and other information that it sees repeatedly. The world knowledge stored in the VLM as a result of pretraining on a vast amount of data means that decisions can be made with information that has a very slim but nonzero chance of showing up in an environment such as the road. The massive corpus of data that is used to pretrain the VLM will have many different pieces of data that an ordinary dataset for training an autonomous vehicle may not necessarily have exposure to. This advantage could give this framework a significant advantage over other autonomous vehicle frameworks.

6.3.8 Human-AI Collaboration

Current methods still fall short of true autonomous vehicles. It is necessary for humans to be able to see how the VLM-MPC system for autonomous vehicles is creating and using its control signals. For this purpose explainable AI (XAI) techniques can be used to gain more insight into how the VLM finds the parameters fed into the MPC. Thankfully, MPC controllers are more naturally interpretable and so XAI techniques may not necessarily even be needed in order to understand that component, so long as the concepts behind the output parameters from the VLM make intuitive sense to a human observer.

6.4 Conclusion and Call to Action

The integration of Multimodal Vision Language Models (VLMs) with Model Predictive Control (MPC) represents a significant leap forward in autonomous vehicle technology. This novel framework has the potential to revolutionize how autonomous vehicles perceive, understand, and navigate complex real-world environments.

By leveraging the semantic understanding capabilities of VLMs and combining them with the predictive and adaptive strengths of MPC, there can be autonomous systems that are more context-aware, safer, and more efficient. This approach addresses critical challenges in autonomous vehicle control, such as real-time decision making in dynamic environments, handling of unforeseen scenarios, and improved interaction with human drivers and pedestrians.

However, realizing this vision requires concerted effort from the research community. I call upon researchers, engineers, and practitioners in the fields of artificial intelligence, robotics,

and autonomous systems to:

1. **Explore VLM-MPC Integration:** Investigate novel ways to effectively combine the strengths of VLMs and MPC, focusing on real-time performance and safety guarantees.
2. **Advance VLM Capabilities:** Develop more efficient and task-specific VLMs that can operate within the computational constraints of autonomous vehicles.
3. **Enhance Safety Frameworks:** Create robust safety validation methods that can rigorously test and verify VLM-enhanced autonomous vehicle control systems.
4. **Address Ethical Considerations:** Engage in interdisciplinary research to tackle the ethical implications of using AI for critical decision making in autonomous vehicles.
5. **Standardize Evaluation Metrics:** Develop comprehensive benchmarks and evaluation frameworks that can assess the performance of integrated VLM-MPC systems across diverse scenarios.
6. **Foster Collaboration:** between academia, industry, and regulatory bodies to accelerate the development and deployment of these advanced autonomous systems.

By pursuing these research directions, the full potential of VLM-enhanced autonomous vehicles can be unlocked, paving the way for safer, more intelligent, and more adaptable transportation systems. The fusion of VLMs with MPC not only promises to advance the field of autonomous vehicle control but also opens up new possibilities for intelligent autonomous agents across various domains.

7.1 Multi-Agent AI for Climate Conservation: Mitigating Deforestation

Multi-Agent Reinforcement Learning (MARL) is useful to model climate conservation problems as Stackelberg Security Games (SSGs) which is elaborated in Section 8.4.5. I have published research [22], focusing on the climate conservation problem of preventing deforestation with the goal of learning a MARL environment that can lead to exploring effective strategies to mitigate deforestation. A boosted Decision Tree Classifier, the BoostIT algorithm constructs the MARL environment modeling a green security asset of the sub-national regions of Indonesia, tree cover. A key insight at the basis of the proposed models is that deforestation occurs in hotspots, according to a thorough literature survey. The strongest predictor of deforestation in other sub-national regions is proximity to other anthropogenic activity. Coarse-grained prediction of targets vulnerable to attacks is a significant challenge in Green Security Games for strategizing by defenders. It takes minimum resources to build a boosted Decision Tree Classifier with accurate predictions, and is scalable for the sake of expanding on the assumptions made regarding the drivers of deforestation. Communities can be empowered by such an algorithm to effectively manage forest resources.

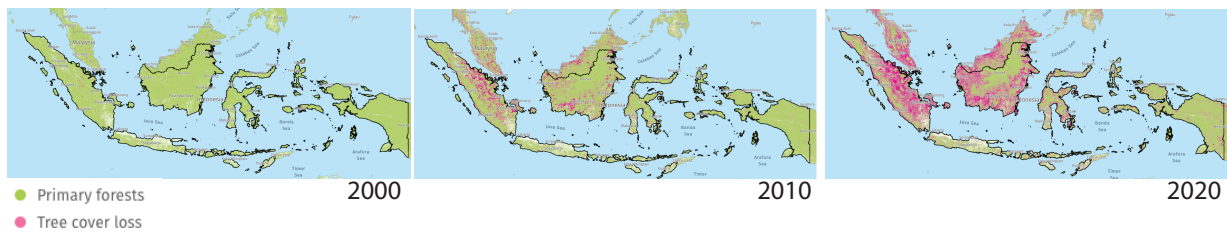


Figure 7.1: Tree cover loss in Indonesia from 2000 to 2020.

Over the last 20 years, the severely vulnerable status of forest assets, specifically in Indonesia, has put the focus on tree cover loss. It can be seen in the image in Figure 7.1 there was a minor tree cover loss in 2020 just 20 years ago. A moderate loss is shown from 2010 in the Western islands of Sumatra with not much change in the remaining islands of Indonesia. However, the tree cover loss rapidly started increasing from 2010- 2020.

7.1.1 Preventing Deforestation: Modeling and Prediction of Vulnerabilities in Forest Conservation

Accurate detection and prediction *hotspots* of vulnerable forest assets is a necessary but challenging task in the domain of *green security* problems i.e., protection of natural assets subject to strategic adversaries [see, e.g., 96]. Solutions in this space are applied towards the management of limited forest resources and, ultimately, the health of our planet. Literature suggests that government bodies, who are tasked with managing resource-rich forests, rarely have comprehensive data regarding deforestation in their region [12]. Reasons include the difficulty of accurate data collection for those protecting green assets, and the vast quantity of data—both in terms of its temporal and spatial depth—can be overwhelming for the many stakeholders involved.

However, data collection has been made much easier with satellite imagery. For example, the Thematic Mapper Plus (ETM+), onboard Landsat 7, and the Operational Land Imager (OLI) onboard Landsat 8 have been systematically acquiring global multi-spectral observations at a 30m spatial resolution¹, since 2013. In the present work, I leverage the data collected repeatedly over an 8-day period by the GLAD lab at the University of Maryland, College Park, to monitor forest activity at a high granularity and accuracy.

This data, openly available through the Global Forest Watch (GFW) initiative, has been enriched using *in-situ* data—collected, on the ground, by environmental scientists. Through this work, I can recognize and map the satellite activity to different types of human activity, contributing to forest cover loss [146]. This work, as seen in Figure 7.1, presents detailed and comprehensive forest data in a simple way.

Stakeholders can be empowered by such predictive models by pointing them to the nexus of human-driven deforestation. I focus on applying predictive models specifically to the Global Forest Watch² data from Indonesia. The reason why I have considered Indonesia was because deforestation is occurring on a massive scale there—ripe for modeling. Figure 7.1 shows that deforestation in Indonesia has accelerated in severity in the last 20 years. This data is readily available. Upward movements of 5 different drivers of forest loss have been detected with satellite imagery alone in previous work [12].

I view deforestation as *adversarial behavior* against forest assets by attackers in a green security game framework [97]. This lens has enabled me to model deforestation as it occurs—much like criminal activity—at *hotspots* which shift over time. I view “attackers” as individuals

¹<https://glad.umd.edu/dataset/glad-forest-alerts>

²<https://www.globalforestwatch.org/>

who extensively consume forest assets and “defenders” as those individuals who work to preserve those same assets. Thus, I believe that my research can be applied towards modeling nature (along with forest rangers) as defender(s) wherein climate change realities disincentivize attacks to forest assets.

I see that models are required to predict the extent of demand for available resources and the growth rate of renewable assets. These model predictors could encourage stakeholders for implementation of forest-farming practices; when they reap only the assets which meet current needs thereby reducing waste. In generating fast, efficient, and detailed prediction models for the short term and long term, I support the development of better mechanisms to empower all people to consider the actual cost of exploiting their resources.

In this Chapter, the focus has been put on specifically predicting the vulnerable areas pertaining to green assets. In order to enhance the decision making in green security problems these predicted vulnerabilities can be used as input data. To provide insight into curbing forest loss, a predictive model has been proposed. The tree-loss-cover data collected over a 20-year period has been used to define attacks on forest assets. A net-positive percent of tree-cover-loss in an area categorizes the area as being attacked. Based on a map of attacks within different sub-national regions in Indonesia, a spatially-aware boosted decision tree model with iterable learning (BoostIT) has been built to predict hotspots vulnerable to deforestation by attackers. Data regarding geographic parameters (elevation in particular) have been used to boost the classification to determine the accessibility of particular regions. I have evaluated the performance of the pure decision tree model (with and without iterable learning), and the performance of the decision tree model with geographic parameters (with and without iterable learning). I have found that all the four proposed classifiers have performed well and have given me some idea of which areas are most

vulnerable to deforestation.

7.1.2 Related Work

I observe that environmental science and mechanism design literature addressed the climate conservation, specially deforestation problem-space most directly. In the following sections, The findings from both fields have been combined to make a case for the type of solution prototype.

Drivers of Deforestation in Indonesia

A large-scale analysis of geographic images has been conducted by the three leading studies [12, 146, 314] to identify deforestation trends. Retrospective land-cover analysis has been conducted by all three projects to correlate in-situ data with satellite imagery to identify drivers. A study was performed in 2018 [314] specifically identifying the drivers in the protected areas of Indonesia. They observed a high correlation between the tree cover loss in protected areas and agricultural expansion. In Indonesia, fires started by farmers near protected lands often unintentionally destroyed protected peat forests. The growth of scrubland, detected on satellite images, followed by forest loss, confirmed this theory. In fact, this study found that more than 38% - 57% of all protected areas which were deforested were left to become scrubland or grassland.

Similar techniques have been used in [12] to understand better the drivers of deforestation in all of Indonesia. They observed that between 2001 and 2016, the single largest contributor was palm oil plantations, accounting for 23% of all deforestation across the country. Deforestation from the oil palm industry hit a peak between 2008 and 2009 and has been steadily declining. Other drivers identified in [12] ordered from highest to lowest impact are: timber industry, unin-

tentional conversion of forests to grassland, small-scale agriculture, small-scale clearing, mining, and fish ponds.

In the latest study [146], a unique deep-learning technique was introduced to automatically identify when deforestation is taking place and its causes. Using expertly labeled satellite images the ForestNet learned to identify drivers. The main thrust of this work is not in identifying unique drivers but in demonstrating that an automatic approach can be employed to identify when these drivers are affecting changes in Indonesian forests. However, it is unlikely that this approach has real-world applications since the images being labeled are static and a proper analysis of human drivers requires observation of trends. Nevertheless, it is a start for those who want to build models to provide more granular predictions.

Note that the studies I discuss above focus on identifying human-made drivers of forest loss and not on natural causes. Statistically, differentiating between the two is considered straightforward. Deforestation caused by anthropomorphic activity can be highly devastating, whereas natural causes do not register on satellite images on nearly the same scale. [12] stratify and filters out forest loss under a certain threshold, and I will be taking the same approach to identify strata of loss.

Preventing Deforestation with Green Security Games

Game theory consists of mathematical modeling among rational decision-makers (as per conventional wisdom) for cooperation, conflict, and other domains of problem-solving [224]. The problems in Green Security Games [153] focus on modeling interactions among agents involved in green security domains like wildlife and biodiversity protection.

Green Security Games can be defined as a game between attackers and defenders on par-

ticular targets holding high significance in green security with coverage vectors indicating if the targets have been covered [226]. There can be complex versions of this game played over several rounds in order to achieve equilibrium.

In the case of wildlife protection [153], the attackers are poachers trying to poach animals like elephants while defenders are the rangers of a protected National Park in Uganda, Africa. Modeling the adversary's decision-making process can be very challenging, primarily to understand the vulnerable areas where an adversary can hunt down targets.

A few different projects are built on this premise and applied towards predicting illegal logging [150, 207], and poaching [333]. These works primarily focus on finding the optimum patrol/defender strategy to prevent illegal extraction of forest resources either by using multi-arm bandits [333], neural networks, or by modeling it as a Stackelberg game [207] [150]. Instead of focusing on optimum defender strategy, I focus on a smaller part of this problem by finding and predicting the most vulnerable forest assets based on previous forest cover loss data and other domain features.

Impact of Multi-Agent AI Strategies on Deforestation

I believe that further work in this research area can empower local stakeholders to manage forest resources and make it the responsibility of the international community— as they also reap the benefits of these resources. I have found quite a few papers in environmental science and economics that proposed solutions towards this end. One solution would be to cap and enforce carbon emissions of international countries tasked with managing resource-rich tropical forests [346]. This approach would limit the amount of deforestation those countries could leverage towards economic gain. However, this can also cause countries to refuse to play in that market—

particularly if they think they can leverage their resources for monetary benefits elsewhere.

It seems that the real issue is that communities tasked with caring for rich forest resources are often at an economic disadvantage in international markets. They find that it is advantageous to sell their resources to powerful multinationals to elevate the status of their country and its people. Thus, mechanisms applied towards reflecting the actual cost of using these resources, be it at the local or global scale, must elevate the voices and power of these local communities [109]. This means providing detailed information to local communities regarding the resources available on their land and the best way to harvest those resources without creating permanent damage to the forest itself. I have found lots of literature that confirmed these ideas and proposed mechanisms to that end. For example, a few proposed enabling a community forestry scheme to allow local communities to directly manage forest resources [1, 262]. [218] proposed, in a nature article, a scheme towards bettering the management of existing plantations in order to reduce deforestation.

To this end, algorithms which reflect these values must be: (1) transparent, (2) readily accessible to local communities, and (3) include information on all the facets which influence deforestation. Details could include: availability of specific resources in a region, the accessibility of those resources across all regions, how these resources are actually valued in the international market, and the risks involved (to bio-diversity or the health of the forests) in harvested those resources.

The ease with which I can train a decision tree classifier and the extensibility of this approach warrant an exploration into using decision tree classifiers towards empowering community-driven forest-management initiatives.

Detecting deforestation

Two of the surveyed papers sought to predict the land areas which were susceptible to deforestation [111, 155]. [155] developed a spatial-temporal analysis technique to identify the primary drivers of deforestation in the Saranda forest of India. They used data within a *GIS framework* to make predictions about the land segments which were the most susceptible to deforestation into the year 2050. Using a combination of analysis techniques, including frequency ratio, logistic regressions, and an analytic hierarchy process, they showed that the susceptible areas were the most likely to be in proximity with settlements or current anthropomorphic activities. This finding is consistent with the literature, which models poaching or illegal logging behavior using *crime hotspots* [150, 207, 333].

[111] observe and predict trends specifically in Indonesian New Guinea. They study the relationship between the proximity of forests to new or existing roads and accessibility based on land properties (slope, elevation, and cost distance to public roads). Weights on existing evidence and logistic regression are used to generate a map that demarcates the areas susceptible to deforestation.

Though my study and ultimate aim are similar, I take a different path in the MARL environment construction by predicting the extent of deforestation in sub-national regions all over Indonesia. This goal can be achieved by ultimately modeling *hotspots* of forest activity, finding that proximity to previous anthropomorphic activity drives current trends.

The [153] thesis introduces a spatially-aware BoostIT (boosted decision tree with an iterative learning algorithm), that is used in the deforestation mitigation modeling. Their thesis uses crime hotspots with soft boundaries in the decision space for less fine-grained segmentation and,

therefore, more capable of representing hotspots to protect wildlife. I compute distance among hotspots with reduced forest cover like their distance function does. They use a parameter vector λ to get the importance of all features, a parameter ω to measure the impact of domain features on observation probability and a β parameter to estimate detector efficiency which is relevant for my model.

7.1.3 Dataset

I have acquired this dataset through the Global Forest Watch Initiative. The data between 2000–2013 was collected and verified, in blocks, by [132]. Since 2013, the global multi-spectral observations were collected by Thematic Mapper Plus (ETM+) onboard Landsat 7, and the Operational Land Imager (OLI) onboard Landsat 8 at a 30m spatial resolution.³ These images were used to produce the tree cover dataset—a product of the University of Maryland’s GLAD lab and Google. They enriched image recognition with *in-situ* data—collected, on the ground, by environmental scientists. Additionally, I have computed terrain features like slope using the Google Earth Engine Python API.⁴ Elevation data from the Shuttle Radar Topography Mission (SRTM) [98] was processed and made accessible by NASA JPL at a resolution of 30m.

I compute the forest loss data from available Forest Change Data over ten years and the percentage of land deforested compared to the total land area for districts in a state. The percentage of deforested land determined whether forests are being *attacked* or not. A detailed description of the parameters, labeling methods, and the generation of the train-test data-splits as inputs for model construction is provided below.

³<https://glad.umd.edu/dataset/glad-forest-alerts>

⁴<https://github.com/google/earthengine-api>

Model	Accuracy	Vulnerable		Not Vulnerable	
		Precision	Recall	Precision	Recall
Base model	62%	72%	76%	27%	23%
Base model with BoostIT	67%	79%	77%	35%	32%
Base model with terrain features	69%	77%	79%	51%	48%
Base model with terrain features and BoostIT	73%	80%	83%	59%	55%

Table 7.1: Performance of different versions of the deforestation vulnerability detection model on the test data.

Parameters and Features

I have considered some basic attributes as parameters for my model. Some of these attributes were unique to Indonesia, and some are general to measuring hotspots of deforestation in any Green Security domain. The deforestation dataset contained five major aspects of deforestation and climate change, namely:

- tree loss cover measured in hectares over 20 years
- biomass loss measured in metric tonnes over 20 years
- CO₂ emissions measured in metric tonnes over 20 years
- different densities of canopy cover measured by the percentage of the tree cover over the land
- and sub-national regions in Indonesia which are analogous to states and districts. (henceforth 'subnational1' and 'subnational2' are referred to as a state and a district respectively)

Tree cover loss data has been critical to label sub-national regions vulnerable to deforestation. Tree cover is defined as all vegetation greater than 5 meters in height. It can take the form of natural forests or plantations across a range of canopy densities. Biomass and CO₂ emissions

information were calculated by Hansen et al. They correlated in-situ data with image detection techniques to get these attributes. Thresholds of canopy cover ($\geq 10\%$, 15% , 20% , 30% , 50% , and 75%) measure the density of trees in a sub-national region. Given a particular area, represented as a circle, the percent canopy cover statistic relates to the percentage of the canopy's ground area. I have looked at data produced between 2000 and 2020 across all canopy densities and all sub-national regions.

Based on existing literature and the data available to us, I have considered and used the set of features to build a spatially aware boosted decision tree model, with iterable learning (BoostIT), to predict hotspots vulnerable to deforestation by adversaries. I have boosted the classification using data regarding geographic parameters (elevation in particular) to determine the accessibility of particular regions. I would like to acknowledge that this is a rough estimate of accessibility. In addition, I have considered metric tonnes of CO₂ emissions and metric tonnes of loss in biomass to reinforce that an attack had taken place. My learned prediction model specifically aims to predict vulnerable regions susceptible to palm tree-related land clearing ([12]).

Labeling Tree Cover Loss as Attacks I have computed the forest loss data from available Forest Change Data over ten years and computed the percentage of land deforested compared to the total land area in 'subnational2' (henceforth referred to as district) in a 'subnational1' region (henceforth referred to as state). The percentage of deforested land determines whether forests are being *attacked* or not, which will be the dependent variable.

If the percentage of deforested land is negative, it indicates that afforestation has happened in that district. Therefore, it is assigned a class label of 0, indicating that the forests in that district for the particular canopy threshold are not vulnerable to deforestation. If the percentage of

deforested land is positive, it indicates that deforestation has happened in that district; therefore, it is assigned a class label of 1 indicating that the forests in that district for the particular canopy threshold are vulnerable to deforestation. For reliability, the annotation of data for targets vulnerable to deforestation must be professionally done by climate change forest experts. Here, they have been automatically labeled for analysis on a low scale with limited resources.

Creation of Training and Test Data In the original dataset, there were 3514 data elements, comprising of 502 districts with seven canopy thresholds. Overall there are 33 distinct states in the datasets. I refer to each data element based on ('subnational1', 'subnational2') keys equivalent to (state, district) as two states can have the same district name.

I have split the available Indonesian forest loss data from Global Forest Change in an 80/20 split to training datasets and test datasets by ensuring that the distribution of the class labels (24% negative labels and 76% positive labels) in the original dataset is maintained in both the train and test split. Suppose data is taken for a (state, district) with a particular canopy threshold in the training file. In that case, data for that same (state, district) for the remaining six canopy thresholds are also placed in the training dataset. The same step is repeated for test data. This is done to have a proper separation between test data and training data.

7.1.4 Methods

I have implemented a version of the spatially aware BoostIT algorithm, introduced by [153] to identify areas that are vulnerable to forest loss. I view loggers as attackers of forests. The goal for this green security domain is to protect the green assets from attackers. Unlike wildlife, which can be targeted in protected areas like national parks, trees can be anywhere. Thus, the notion

of defenders in this green security game can and should be broad. I will recommend probable defenders who can effectively act on information on vulnerable areas prone to deforestation, subject to incentives.

Initially, I have trained a Decision Tree Classifier to predict areas vulnerable to tree-cover loss using features from the original dataset. I have improved it via the BoostIT decision tree algorithm as outlined in Algorithm 3. I have predicted hotspot labels on training data and test data.

Next, to calculate hotspot proximity, I have first devised Algorithm 4 to compute a notion of proximity in the dataset. The simple idea of proximity is that all districts in a state are proximate, but districts in different states are not. This algorithm considers predicted hotspot class labels from training data (Θ^h) or test data (Ψ^h) as inputs. Then it finds their state ids denoting whether they are close to one another. Then for each district, the number of districts close to it is computed. If this number is above a certain threshold α , a new spatial feature (h) is added to the dataset updating the training data (Θ^h) or test data (Ψ^h) accordingly—thereby learning the boosted decision tree. α is set to 5 in the experiments. This was repeated for a fixed number of iterations (10 times in my experiment). I have used the original decision tree and the ten boosted decision trees for accuracy calculation and further analysis.

Please refer to Section 7.1.4 for details regarding hotspot proximity calculation and to see the visualization of the learned boosted decision tree model (Figure 7.3).

Hotspot Proximity Calculation

To calculate hotspot proximity, I have first devised an algorithm to compute a notion of proximity in the dataset as outlined in algorithm 4. The simple idea of proximity is that all

Algorithm 3 BoostIT algorithm to detect targets vulnerable to deforestation

Input: *train_data*, *test_data*, *proximity_vector*, *iterations*

Parameter: *alpha* = 5

Output: *trees*, *gtrlist*, *gtelist*

```
1:  $\Theta_0 \leftarrow \text{train\_data}$ 
2:  $\Psi_0 \leftarrow \text{test\_data}$ 
3:  $D_0 \leftarrow \text{learn\_decision\_tree}(\Theta_0)$ 
4:  $gtr_0 \leftarrow \text{predict\_labels}(D_0, \Theta_0)$ 
5:                                     ▷ Predict labels on training data
6:  $gte_0 \leftarrow \text{predict\_labels}(D_0, \Psi_0)$ 
7:                                     ▷ Predict labels on test data
8:  $i \leftarrow 0$ 
9: trees ← initialized as list with 1st element as  $D_0$ 
10: gtrlist ← initialized as list with 1st element as  $gtr_0$ 
11: gtelist ← initialized as list with 1st element as  $gte_0$ 
12:  $\Theta_0^h \leftarrow \Theta_0$ 
13:  $\Psi_0^h \leftarrow \Psi_0$ 
14: while  $i < \text{iterations}$  do
15:   htrain_name ← "h_train_iteration" + str(i)
16:    $\Theta_i^h \leftarrow \text{calc\_hotspot\_prox}(gtr_i, \Theta_i^h, \text{htrain\_name})$ 
17:                                     ▷ Spatial feature is added to training features
18:   htest_name ← "h_test_iteration" + str(i)
19:    $\Psi_i^h \leftarrow \text{calc\_hotspot\_prox}(gte_i, \Psi_i^h, \text{htest\_name})$ 
20:                                     ▷ Spatial feature is added to test features
21:    $D_i \leftarrow \text{learn\_decision\_tree}(\Theta_i)$ 
22:    $gtr_i \leftarrow \text{predict\_labels}(D_i, \Theta_i)$ 
23:                                     ▷ Predict labels on training data
24:    $gte_i \leftarrow \text{predict\_labels}(D_i, \Psi_i)$ 
25:                                     ▷ Predict labels on test data
26:   trees.append( $D_i$ )
27:   gtrlist.append( $gtr_i$ )
28:   gtelist.append( $gte_i$ )
29:    $i \leftarrow i + 1$ 
30: end while
31: return trees, gtrlist, gtelist
```

districts in every state are proximate to one another, but districts in different states are not. This oversimplification may be pertinent in Indonesia, with thousands of islands formed states based on proximity. Further research can be done to calculate the exact distance between districts to make bordering districts in adjoining states proximate to one another.

The proximity vector is computed, which is an array of 3514 elements as there are 3514 data points. Each element is the state id for the particular district. The notion of proximity as outlined in algorithm 4 is essential to calculate hotspot proximity in algorithm 5 which is used in the BoostIT algorithm.

This algorithm considers predicted hotspot class labels from training data (Θ^h) or test data (Ψ^h) as inputs. Then it finds their state ids denoting whether they are close to one another. Then for one district, the number of districts close to it is computed. If this number is above a certain threshold α , a new spatial feature (h) with value one is added to the dataset updating the training data (Θ^h) or test data (Ψ^h) accordingly. α is set to 5 in the experiments.

Algorithm 4 Proximity Calculation

Input: *orig_data*

Output: *proximity_vector*

```

1:  $l \leftarrow \text{length}(\text{orig\_data})$ 
2:    $\triangleright$  orig_data has features for every canopy threshold in every district of every Indonesian state
3:  $\text{proximity\_vector} \leftarrow$  initialized to an array with 0s having length  $l$ 
4:  $\text{unique\_states} \leftarrow \text{orig\_data}[\text{'subnational1'}].\text{unique}()$ 
5:    $\triangleright$  Gets a list of Indonesian states
6:  $\text{state\_id} \leftarrow 0$ 
7: for each_state in unique_states do
8:    $\text{index\_all\_districts\_in\_a\_state} \leftarrow$  list of all indices of each_state in orig_data
9:   for index in index_all_districts_in_a_state do
10:     $\text{proximity\_vector}[\text{index}] = \text{state\_id}$ 
11:   end for
12:    $\text{state\_id} \leftarrow \text{state\_id} + 1$ 
13: end for
14: return proximity_vector

```

Algorithm 5 Calculate Hotspot Proximity

Input: *predictions, feature_name, data***Parameter:** $\alpha = 5$ **Output:** *data*

```
1: hotspot_indices  $\leftarrow$  gets a list of indices in data for hotspots (class label = 1) in pred
2: hotspots  $\leftarrow$  state_ids from proximity_vector at hotspot_indices
3: hotspot_state_counts  $\leftarrow$  a 2D array with unique states and their counts in hotspots
4: data[feat_name]  $\leftarrow$  0
5:                                      $\triangleright$  A new column initialized to 0 is added to the data
6: for state_count in hotspot_state_counts do
7:   state = state_count[0]
8:   count = state_count[1]
9:   if count  $\leq$   $\alpha$  then
10:     dist_indices_cutoff  $\leftarrow$  indices from prox_vec for state_ids with value state
11:     data[dist_indices_cutoff, feat_name] = 1
12:      $\triangleright$  Sets the new feature in data as 1 for elements in data at dist_indices_cutoff
13:   end if
14: end for
15: return data
```

7.1.5 Results

The results on the comparison between different models are presented in Table 7.1.⁵ I can see that all four models have decent performance (accuracy > 62%) and can give some idea of which areas are most vulnerable to deforestation. The mismatch in the precision and recall between the two labels can be explained by the fact that even if two locations have the same features, they could be labeled differently. Boosting the decision tree with proximity lead to significant improvement in prediction results—both for the base model and the base model with the addition of terrain features (see Figure 7.3). This approach aligns with the intuition that identifying areas vulnerable to deforestation is more akin to detecting hotspots than segmentation. The addition of terrain features also leads to a significant jump in performance. Biomass features

⁵I have defined each model and included the confusion matrix of the boosted base model with terrain features in Figure 7.2.

seem to be the most important for my base model. However, the slope and biomass standard deviations for the base model, with terrain features, were equally essential features.

I evaluate the following four versions of my AI model on the test split of the created dataset in order to understand how the assumptions on environmental and geological features affect the performance of the model :

- **Base model:** This model is composed of a decision tree used as a classifier.
- **Base model with BoostIT:** This model is composed of a decision tree along with the BoostIT algorithm applied to it as shown in Algorithm 3.
- **Base model with terrain features:** This model is composed of a decision tree with the inclusion of terrain features (e.g. slope).
- **Base model with terrain features and BoostIT:** This model is composed of a decision tree with the inclusion of terrain features (e.g. slope) and also with the BoostIT algorithm applied to it as shown in Algorithm 3.

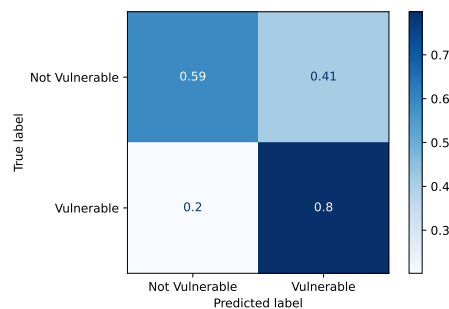


Figure 7.2: Confusion matrix for the base model with terrain features and BoostIT

The results on the comparison between different models are presented in Table 7.1. I have also included the confusion matrix for the base model with terrain features and BoostIT in Figure 7.2.

Looking at the results, I can see that all four models have decent performance and can give some idea of which areas are most vulnerable to deforestation. The mismatch in the precision and recall between the two labels can be explained by the fact that even if two locations have the same features doesn't mean that they would have the same label. There are many other factors that my simple model doesn't take into account. For example, maybe one of the areas has a significant portion of the forest classified as protected land by the government. This would make this area a lot less vulnerable to deforestation.

Another interesting thing to note is that boosting the decision tree with proximity leads to significant improvement in prediction results both for the base model and the base model with the addition of terrain features. This is in line with the intuition that identifying areas vulnerable to deforestation is more akin to detecting hotspots rather than segmentation.

The addition of terrain features also leads to a significant jump in performance. One interesting thing to note is that while the performance for samples with Vulnerable

Since the decision tree is an interpretable model, I have also analyzed which features were most important for the model. The importance of a feature is defined as the reduction in the loss function brought about by adding that feature. For the base model, the biomass feature was most important while for the base model with terrain features added to it, the standard deviation of the slope and biomass features were the most important features.

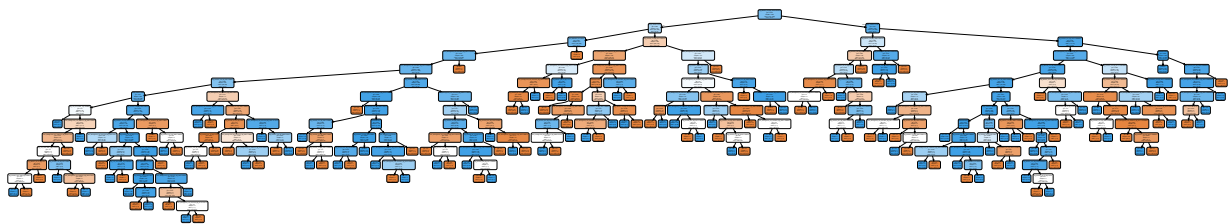


Figure 7.3: Visualization of the decision tree learned for the base model with terrain features and BoostIT (Please zoom in to see the individual nodes of the decision tree more clearly)

7.1.6 Conclusion

The BoostIT shows improved precision, recall, and accuracy metrics compared to a baseline decision tree model for the task of predicting vulnerable deforestation hotspots. When the BoostIT algorithm is implemented, the precision of a basic decision tree improves from 72% to 79%. Considering other terrain features like slope further boosts performance to 80%, suggesting interesting patterns, such as steeper regions being more vulnerable because the logs can be easily pushed down a hill. To improve accuracy and precision even more, my BoostIT decision tree algorithm could be extended a version of spatial XGBoost(Extreme Gradient Boosting) [64].

The results show the addition of fine-grained features is essential to accurately predicting regions susceptible to forest cover loss. This implies automatically generating reliable data for green security games is a very achievable goal.

This approach shows that even rudimentary models are capable of accurately and efficiently predicting regions vulnerable to deforestation. The algorithm is easily generalizable to any region worldwide that has available forest-cover data.

The green security game and other game-theoretic models are only as reliable as the data they are based on [97]. I address this in the pipeline's first stage by estimating metrics for uncertainty in the vulnerability predictions. This enhances the decision making process for the defenders within the green security game [69].

Potential avenues for future work include integrating additional fine-grained features from Google geo-location services and satellite images such as the proximity of vulnerable hotspots to human settlements, rivers, roads, and farmland. The variance of the data once these features are considered will need to be addressed. There are more features that will be hard to obtain, but

will also have an out-sized impact on forest cover, such as who owns the land and how dense the forest is in each area.

The features are also dynamic. As an example, topography can change due to natural disasters and new roads can be built. Forest loss is considered at a granular district level for the precise identification of deforestation hotspots. As a result, it is more feasible to implement concrete strategies for protecting vulnerable hotspots.

Modeling the detection of hotspots vulnerable to deforestation through green security games can be the foundation for the management and protection of forest resources. Defending an entire forest from illegal logging and managing those forest resources is a widespread, pervasive and global problem, very different from the task of protecting forest assets in a designated conservation area.

7.2 AI Agents to Autonomously Orchestrate in Supply Chains

The global economy is reliant on interactions of multiple economic agents to orchestrate the movement of goods worth trillions of dollars across the multi-echelon global supply chain. It is important to plan warehouse inventory management strategies resiliently to address emerging challenges in global supply chains like a sudden pandemic-related shutdown or address regular issues like seasonal fluctuations in consumer demand, dwell time of goods in warehouses, and cold-chain shipments with perishable goods. Efficient inventory management strategies can reduce consumer prices, maximize warehouse profits, and effectively serve market demand. Existing research uses Multi-Agent Reinforcement Learning (MARL) to autonomously learn replenishment strategies from hierarchical interactions of warehouse agents in non-stationary multi-echelon in-

ventory environments. However, this can lead to unstable training on significantly increasing the scale of economic agents, along with sub-optimal performance under limited resources, challenges in cooperation along the multi-echelon inventory management paradigm, robustness, and generalization issues.

The Seasonal Trends for Efficient Inventory management with Multi-Agent Coalitions (STEIMAC) framework is built on top of existing MARL benchmarks, adding new temporal features like seasonal demand fluctuations affecting warehouse inventory capacities during months of less demand slowing down the supply chain, dwell time in warehouses leading to price stagnation or drop in commodity prices decreasing inventory profits, and cold-chain products which can be perishable high price products like vaccines or low price products like grocery food. Stable strategies for inventory management, leading to more profits across multiple episodes, can be learnt using off-policy Multi-Agent algorithms like QMIX and Weighted QMIX in comparison to on-policy Multi-Agent algorithms like IPPO or MAPPO, while also learning outlier economic trends. The STEIMAC framework is significantly faster to learn policies by allowing the economic agents in warehouses to form coalitions based on the seasonal and perishable nature of the goods they're storing and trading and also considering the dwell time of goods. This is because the exponential action space complexity for multiple agents in MARL is highly reduced when the economic agents can form coalitions of agents performing similar actions. The ability to learn efficient and stable policies to manage multi-echelon inventory stocks has led to improving the profit margins by saving unnecessary interactions of inventory agents with reduced dwell time addressing seasonal changes in consumer demand.

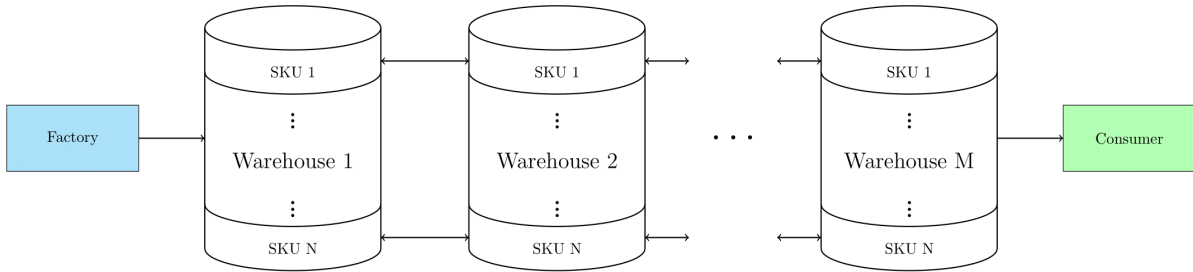


Figure 7.4: Illustration of the bus network setting. Each agent represents an SKU in a warehouse, with interactions occurring between corresponding SKUs across warehouses. Agents’ actions involve requesting specific replenishment quantities from upstream warehouses, while warehouses sell a portion of each SKU’s inventory to downstream warehouses based on the replenishment order.

7.2.1 Introduction

The multi-echelon global supply chain is crucial to orchestrate the movement of goods worth trillions of dollars where the resilience of inventory management strategies become important to reduce prices for consumers, maximize profit, and serve the demand in markets. However, there are significant challenges in this multi-echelon inventory management problem involving large scale interactions of economic agents in different hierarchies of the inventory management problem who have to make optimal decisions that are helpful to replenish the inventories in the short term while maintaining long-term profit considering warehouse capacities, vendor lead times, procurement costs, demand, cost of goods overflowing the warehouse among other factors.

Multi-Agent Reinforcement Learning (MARL) is useful to model the interaction of economic agents in the autonomous management of the multi-level inventory. This is demonstrated by the Multi-Agent Benchmark for Inventory Management (MABIM) [335], where each agent can be considered to be making decisions for each Stock Keeping Unit (SKU) in a warehouse, procuring and selling items to replenish the inventory. A comparative analysis with Operations

Research (OR) algorithms leads to insights into challenges of current MARL approaches having unstable training on significantly increasing the scale of agents, performing sub-optimally under limited resources, challenges in cooperation along the multi-echelon inventory management paradigm along with robustness and generalization issues.

This setting, illustrated in Figure 7.4, is modeled after the MABIM setting. I improve the MABIM benchmark by defining new temporal parameters like seasonality to effectively model interactions of economic agents for inventory management. Seasonality involves understanding the trends of demand and procurement over years, months, dates, and days. There may be seasonal increase in purchase patterns during the Holiday Months in Summer and Winter while purchase patterns can stagnate during weekdays when consumers are busy at work. Learning seasonal purchase patterns can be helpful for economic agents to better strategize their plans for replenishing the Stock Keeping Units (SKUs). If the selling price stagnates and demand in the same period also stagnates, then the goods need to be sold off quickly depending on the capacity of the warehouse and accordingly, the replenishment policies get updated. Seasonality is a distributional risk dependent on the data in the corresponding time period. During simulation, seasonal change of demand has a similar trend across seasons over a few years unlike cold chains, whose accessibility/inaccessibility determine whether seasonal change is not important while cold chain's impact on the supply chain keeps getting updated over-time. STEIMAC significantly expands the variety of algorithms compared to MABIM, as seen in Table 7.2.

I also include the ability of the different economic agents to trade on different items instead of allowing trading on one kind of item as encoded in the MABIM benchmark. For my simulation benchmark, I provide randomly sampled synthetic data for an item with a lower selling price compared to the current item and another item with a higher selling price. For the higher sell-

Category	Type	Algorithm	STEIMAC (Ours)	MABIM
MARL	On-Policy	IPPO [85]	✓	✓
		MAPPO [341]	✓	×
	Off-Policy	QMIX [248]	✓	×
		QTRAN [281]	✓	✓
		Weighted QMIX [247]	✓	×
Operations Research		(s,S) Policy [357]	✓	✓

Table 7.2: Comparison of Algorithms Covered by STEIMAC (Ours) and MABIM

ing price item, I categorize it as a cold-chain shipment product, thereby categorizing ten SKUs in two warehouses as capable of storing cold-chain shipments. Thus, I update the network of warehouses interacting with each other where current research considers a warehouse to interact with at most two warehouses, one for procurement and one for selling, in a multi-echelon setup while I add the cold-chain capable warehouses to trade amongst themselves additionally to capitalize on that capacity. The selling price and demand need to be multi-valued only for those SKUs having cold-chain shipment capabilities. I incorporate stochastic uncertainties, weighing my demand with stochastic functions to model the aleatoric risk involved. This could manifest as increase in demand, even during weekdays, which may be possible due to stochastic events like the COVID-19 pandemic [15] when consumers had to work from home and accordingly saved time to purchase more. This lead to an increase in demand with respect to similar time periods of purchase. Another example of stochastic risks can involve product recalls where the warehouses may have to sell off a product anticipating recall, irrespective of it's widespread demand.

The availability of cold-chain shipment capabilities introduces partial observability in the warehouse where economic agents operating SKUs in one warehouse may not be aware of the cold-chain shipment capabilities of SKUs in other warehouses. I model the inventory management problem as a Dec-POMDP (Decentralized Partially Observable Markov Decision Process)

where the agents in one warehouse share a joint reward and aim to cooperate with agents in other warehouses. I investigate algorithms like Weighted QMIX [247] and MAPPO [341] in addition to the IPPO [85] and QTRAN [281] algorithms as supported by the MABIM benchmark to compute the Multi-Agent reward faster and better. To improve generalizability, I have my MARL algorithms to learn an internal self-supervised reward with Expectation Alignment (*ELIGN*) [199] where the agent's reward matches the expectations of agents' neighbors, tackling challenges of sparse rewards in the modeling of economic decision making in inventory management. *ELIGN* does not work well in complex state spaces like my Multi-Agent supply-chain environment for which I introduce a new CUriosity-driven Reward from Latent Spaces (*CURLS*) which models clusters of state space observation samples to incentivize exploration of new clusters.

The contributions of this work are as follows:

- **STEIMAC:** I propose the Seasonal Trends for Efficient Inventory Management with Multi-Agent Cooperation (*STEIMAC*) framework. *STEIMAC* extends existing Multi-Agent supply chain benchmarks by integrating temporal features like seasonality and cold-chain products into the inventory management problem.
- **Temporal Modeling:** My approach captures trends and seasonal fluctuations into important variables such as pricing and demand using additive seasonal decomposition.
- **Cold-Chain SKU Modeling:** I model the epistemic uncertainty inherent to the storage of perishable goods using a gamma distribution to decay the price of such products over time, accounting for the real world challenge of stocking cold chain items.
- **Intrinsic Rewards:** I encourage cooperation and exploration with two self-supervised rewards: the existing *ELIGN* reward from previous work and *CURLS*, a novel reward that

provides a bonus for observing unfamiliar states.

- **Comprehensive Empirical Evaluation:** I run my experiments under a wide variety of experimental conditions with different scales of SKUs and come to the conclusion that on-policy MARL algorithms (e.g. IPPO, MAPPO) converge better and provide more stable training for multi-echelon supply-chain environments in comparison to off-policy algorithms (e.g. QMIX, WQMIX, QTRAN).

7.2.2 Related Work

The multi-stage inventory management problem can be first attributed to the works of [110, 329], who formalized adding heuristics such as safety stock norms to decrease costs. One of the most important algorithms that solve the inventory management problem is the (s, S) policy, which replenishes the stock to S when the inventory drops below s [149, 310]. However, such heuristics are targeted towards environments where demand and lead times are fixed, so they cannot be applied to real-world scenarios. Since demand is affected by pricing, marketing, and external trends, [61] argue that demand should be modeled as a stochastic process, as [112] assumed log-normal and gamma demand distributions. This allowed for improved heuristics to model the lead time, shortage, etc. to better model the inventory control problem.

Operations Research (OR) Methods for Inventory Control

The multi-stage inventory management problem can be first attributed to the work of Fukuda [110], Wijngaard and Wortmann [329], who formalized adding heuristics such as safety stock norms to decrease costs. One of the most important algorithms that solve the inventory

management problem is the (s, S) policy, which replenishes the stock to S when the inventory drops below s [149, 310]. However, such heuristics are targeted towards environments where demand and lead times are fixed, and cannot be modeled effectively in dynamic real-world scenarios. For example, a retailer managing winter coats might use the (s, S) policy by setting $s = 50$ and $S = 200$, which means that whenever the inventory drops to below 50 units, an order is placed to restock up to 200 coats. However, if demand fluctuates due to weather changes, marketing campaigns, or unexpected fashion trends, this fixed threshold can lead to overstocking (when the demand is too low) or stockouts (when the demand is too high). Since demand is dynamic and is affected by pricing, marketing, and external trends, Chaudhary et al. [61] argue that demand should be modeled as a stochastic process, as Gholami and Mirzazadeh [112] assumed log-normal and gamma demand distributions. This allowed for improved heuristics to model the lead time, shortage, etc. to better model the inventory control problem.

Reinforcement Learning for Inventory Control

While heuristic-based approaches work well for fixed demands, or for the demand that can be modeled by common probability distributions, they struggle for more complex demand distributions. As a result, many studies pivoted towards using reinforcement learning to model the inventory control process due to its ability to adapt to changes in the environment in real time. For example, [115] design a reward system based on the average profit at each state, achieving considerable decreases in total cost compared to baseline periodic order policies [49]. Additional studies using several RL algorithms, such as policy gradients [128], A3C [38], and PPO [128, 236, 330], have had improvements over heuristic methods. Another approach is to use Deep Reinforcement Learning (DRL) by including another set of parameters in the form of a

neural network while training. [317] use a Double Deep Q-Network approach [308] by replacing the Q-table with a neural network, which allows for storing more complex trajectories and for using experience replay to learn from past experiences. In addition, using reinforcement learning allows rewards to be shaped through several different cost functions. For example, [175] sets the reward at each time step t as a linear combination of ordering, holding, and shortage costs at that time step, which directly optimizes the agent to minimize this cost.

Multi-Agent RL in Inventory Management

To extend the multi-echelon inventory management problem to several items, studies have used Multi-Agent RL (MARL) algorithms, where one or more agents can model the inventory of each item. This poses at least two advantages over single-agent RL algorithms. First, agents can act either fully cooperatively, where each agent obtains a joint reward at time step t , fully competitively, where each agent obtains a different reward at time t , or a mix of both approaches [230]. In the inventory management setting, this allows for more scalable algorithms, as larger multi-echelon systems require cooperation between agents for ideal product transfer, and competition between agents improves inventory stocking efficiency when there is limited warehouse capacity or increased storage costs [157, 335]. Second, this allows for decentralized algorithms that can constrain the problem to several independent agents, which can be solved by existing single-agent RL techniques. For example, [89] formulates the inventory management problem as a Shared Resource Stochastic Game, where agents only interact via a shared resource limit and know how many resources all other agents are using. This can be represented as context about all other agents and is used to develop a local simulator that reduces the problem to single-agent RL.

7.2.3 Multi-Agent RL Algorithms for Cooperation in Supply Chains

This work focuses on several Multi-Agent RL algorithms for fully cooperative settings, as the agents represented by each SKU are not competing against one another. Cooperation can be formalized by setting the reward of each of the agents to be the same: $R_1 = R_2 = \dots = R_N$, which is the shared reward from the dec-POMDP formulation. This results in maximizing an expected total reward of

$$J(\pi) = \mathbb{E}_{s \sim \rho_\pi, \mathbf{a} \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \mathbf{R}_t \right]$$

where γ is the discount factor and \mathbf{R}_t is the shared reward between all agents at time t [166].

IPPO

Independent PPO (IPPO) is an on-policy independent learning algorithm that attempts to scale Proximal Policy Optimization (PPO) to multiple agents by applying PPO individually. This means that each agent learns solely by its observations and rewards and treats all other agents as part of the environment, which works for the STEIMAC environment, as each SKU effectively operates independently from other SKUs. Additionally, the independent nature of IPPO allows for highly parallelizable computation. However, IPPO has stability concerns when scaling up the number of agents, as each agent has its own loss functions to minimize. Since PPO is an actor-critic algorithm, the actor loss for IPPO is

$$\mathcal{L}^a(\theta) = \mathbb{E}_{u_t^a, z_t^a} \left[\min \left(\frac{\pi_\theta(u_t^a | z_t^a)}{\pi_{\theta_{old}}(u_t^a | z_t^a)} A_t^a, \text{clip} \left(\frac{\pi_\theta(u_t^a | z_t^a)}{\pi_{\theta_{old}}(u_t^a | z_t^a)}, 1 - \epsilon, 1 + \epsilon \right) A_t^a \right) \right]$$

where π_θ is the current policy parameterized by the actor network θ , $\pi_{\theta_{old}}$ is the policy parameterized by θ from the previous step, and A_t^a is an estimate of the advantage function [85]. u_t^a and z_t^a are used to represent the individual action and state for each agent respectively. The critic loss for IPPO is

$$\mathcal{L}^a(\phi) = \mathbb{E}_{z_t^a} \left[\min \left(\left(V_\phi(z_t^a) - \hat{V}_t^a \right)^2, \left(V_{\phi_{old}}(z_t^a) + \text{clip} \left(V_\phi(z_t^a) - V_{\phi_{old}}(z_t^a), -\epsilon, \epsilon \right) - \hat{V}_t^a \right)^2 \right) \right]$$

where V_ϕ is a learned state-value function parameterized by the critic network ϕ , $V_{\phi_{old}}$ is another learned state-value function parameterized by the old critic network from one step before, and $\hat{V}_t^a = A_t^a + V_\phi(z_t^a)$. Finally, the policy entropy for each agent $\mathcal{H}(\pi^a)$ is included, which is a measure of uncertainty in the policy and acts as a regularization term in the actor-critic loss. Thus, the overall IPPO actor-critic loss can be written as

$$\mathcal{L}^a(\theta, \phi) = \sum_{i=1}^N \mathcal{L}^a(\theta) + \lambda_c \mathcal{L}^a(\phi) + \lambda_h \mathcal{H}(\pi^a)$$

where $\lambda_c, \lambda_h \in \mathbb{R}^+$ are hyperparameters for the critic loss and the policy entropy [85].

QMIX

QMIX is an off-policy action-value function factorization algorithm that learns a joint action-value function $Q_{tot}(\boldsymbol{\tau}, \mathbf{u})$ to develop consistent centralized and decentralized policies. I denote $\boldsymbol{\tau}$ as the joint action-observation history and \mathbf{u} as the joint action. Previous action-value function factorization algorithms, such as Value Decomposition Networks (VDNs), compute the joint action-value function for a centralized policy by taking the sum of each individual action-

value function $Q_i(\tau_i, u_i, \theta_i)$. However, this full factorization is not required in QMIX to maintain consistency between the centralized policy and each decentralized policy, only as long as the $\arg \max$ of the joint action-value function is the same as the $\arg \max$ of each individual action-value function [248].

$$\arg \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \arg \max_{u_1} Q(\tau_1, u_1) \\ \vdots \\ \arg \max_{u_n} Q(\tau_n, u_n) \end{pmatrix}$$

This constraint, called the Individual Global Maximum (IGM), is satisfied by VDNs, as the best action from the sum of the action-value functions is the same as the best action from each action, but is highly restrictive on the possible action-value functions. Instead, QMIX proposes that the condition can be relaxed to any monotonic functions, where monotonicity is represented by

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A$$

The QMIX algorithm enforces monotonicity by using a mixing hyper-network where all the weights are non-negative. This mixing network takes in the Q-values from each action-value function and the current state and outputs the joint action-value function. This mixing network is trained by the temporal difference loss

$$\mathcal{L}(\theta) = \sum_{i=1}^b \left[(y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta))^2 \right]$$

where b batches are sampled from the replay buffer (as QMIX is an off-policy algorithm) and use the target value $y^{tot} = r + \gamma Q_{tot}(\boldsymbol{\tau}', \mathbf{u}', s'; \theta^-)$, where θ^- is a target network for the joint action-value function.

WQMIX

While QMIX allows for a larger range of possible joint action-value functions, there are several problems. Firstly, the QMIX operator $T_{Q_{mix}}^*$ is not a contraction, which prevents accurate convergence [247]. Additionally, there are Q functions that return a different $\arg \max$ than the output of QMIX, which can lead to incorrect actions being selected and an overall worse policy. To mitigate this, Weighted QMIX assigns weights to each of the action-value functions to compute the joint action-value functions, and uses an approximation of the true Q value \hat{Q}^* to help train the network. Weighted QMIX aims to minimize both the Q_{tot} temporal difference loss

$$\sum_{i=1}^b [w(s, \mathbf{u})(Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s) - y_i)^2]$$

and the \hat{Q}^* temporal difference loss

$$\sum_{i=1}^b [(\hat{Q}^*(\boldsymbol{\tau}, \mathbf{u}, s) - y_i)^2]$$

where $y_i = r + \gamma \hat{Q}^*(s', \boldsymbol{\tau}', \arg \max_{\mathbf{u}'} Q_{tot}(\boldsymbol{\tau}', \mathbf{u}', s'))$ for both losses. There are typically two weightings for Weighted QMIX, Centrally Weighted QMIX and Optimistically Weighted QMIX.

Centrally Weighted QMIX uses the weighting

$$w(s, \mathbf{u}) = \begin{cases} 1 & y_i > \hat{Q}^*(s, \boldsymbol{\tau}, \hat{\mathbf{u}}^*) \text{ or } \mathbf{u} = \hat{\mathbf{u}}^* \\ \alpha & \text{otherwise} \end{cases}$$

where $\hat{\mathbf{u}}^* = \arg \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s)$. This uses the approximation of \hat{Q}^* , so if $y_i > \hat{Q}^*(s, \boldsymbol{\tau}, \hat{\mathbf{u}})$, then there is a high likelihood that \mathbf{u} is the best action for this action-value function. Optimistically Weighed QMIX has the weighting

$$w(s, \mathbf{u}) = \begin{cases} 1 & Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s) < y_i \\ \alpha & \text{otherwise} \end{cases}$$

where for both weightings, $\alpha \in \mathbb{R}^+$ and $0 \leq \alpha \leq 1$. For the experiments, I have used Centrally Weighted QMIX.

QTRAN

QTRAN is another off-policy Multi-Agent RL algorithm that introduces a separate constraint to enforce the factorizability of the joint action-value function to mitigate the pitfalls of VDNs and QMIX. QTRAN enforces the factorization through the following constraint:

$$\sum_{i=1}^N Q_i(\tau_i, u_i) - Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) - V_{jt}(\boldsymbol{\tau}) = \begin{cases} 0 & \mathbf{u} = \bar{\mathbf{u}} \\ \geq 0 & \mathbf{u} \neq \bar{\mathbf{u}} \end{cases}, V_{jt}(\boldsymbol{\tau}) = \max_{\mathbf{u}} Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) - \sum_{i=1}^N Q_i(\tau_i, \bar{u}_i)$$

Additionally, QTRAN includes another joint action-value function defined as the sum of individual action-value functions

$$Q'_{jt}(\boldsymbol{\tau}, \mathbf{u}) = \sum_{i=1}^N Q_i(\tau_i, u_i)$$

which is used during training to help enforce the constraint [281]. QTRAN is trained via a temporal difference loss for estimating the true action-value function, a nonoptimal loss that checks whether the condition holds if a nonoptimal action is selected, and an optimal loss that checks whether the condition holds if the optimal local action is selected:

$$\mathcal{L}_{td}(\boldsymbol{\tau}, \mathbf{u}, r, \boldsymbol{\tau}'; \theta) = (Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) - y^{dq}n(r, \boldsymbol{\tau}'; \theta^-))^2$$

$$\mathcal{L}_{opt}(\boldsymbol{\tau}, \mathbf{u}, r, \boldsymbol{\tau}'; \theta) = (Q'_{jt}(\boldsymbol{\tau}, \bar{\mathbf{u}}) - \hat{Q}_{jt}(\boldsymbol{\tau}, \bar{\mathbf{u}}) + V_{jt}(\boldsymbol{\tau}))^2$$

$$\mathcal{L}_{nopt}(\boldsymbol{\tau}, \mathbf{u}, r, \boldsymbol{\tau}'; \theta) = (\min[Q'_{jt}(\boldsymbol{\tau}, \mathbf{u}) - \hat{Q}_{jt}(\boldsymbol{\tau}, \mathbf{u}) + V_{jt}(\boldsymbol{\tau}), 0])^2$$

where $\boldsymbol{\tau}$ is the action-observation history, $\boldsymbol{\tau}'$ is another action-observation history including the next action, \mathbf{u} is the joint action, r is the reward associated with taking \mathbf{u} , $\bar{\mathbf{u}}$ is the optimal action between all individual action-value functions, and θ^- is a target network periodically copied in a fashion similar to DQN [281]. The overall loss for QTRAN is calculated as

$$\mathcal{L}(\boldsymbol{\tau}, \mathbf{u}, r, \boldsymbol{\tau}'; \theta) = \mathcal{L}_{td} + \lambda_{opt}\mathcal{L}_{opt} + \lambda_{nopt}\mathcal{L}_{nopt}$$

where $\lambda_{opt}, \lambda_{nopt} \in \mathbb{R}^+$ are hyperparameters for the optimal and nonoptimal loss respectively.

MAPPO

Similar to IPPO, Multi-Agent Proximal Policy Optimization (MAPPO) is another on-policy algorithm that scales PPO to multiple agents but uses a joint reward to learn stable policies.

7.2.4 RL Algorithms with Base Stock Constraints

I also include base stock variants to IPPO and QTRAN, which were provided by the MABIM dataset [335]. Base stock policies introduce more stability into the learning algorithms by ensuring that the stock level does not fall below a certain level. I can frame the base stock algorithm as a linear programming problem:

$$\begin{aligned}
 \max R_t^{i,j} &= \lambda_r^i s_t^{i,j} - \lambda_p^i s_{t+1}^{i,j} - \lambda_s^i x_{t+1}^{i,j} - \lambda_p^i T_0^{i,j} - \lambda_p^i x_0^{i,j} \\
 s_t x_{t+1}^{i,j} &= x_t^{i,j} + s_{t-v_t^{i,j}}^{i+1,j} - s_t^{i,j} \\
 T_{t+1}^{i,j} &= T_t^{i,j} - s_{t-v_t^{i,j}}^{i+1,j} + s_t^{i,j} \\
 s_t^{i,j} &= \min(x_t^{i,j}, p_t^{i,j}) \\
 T_0^{i,j} &= \sum_{i=-v_t^{i,j}}^{-1} x_t^{i+1,j} \\
 z^{i,j} &= x_{t+1}^{i,j} + s_t^{i+1,j} + T_t^{i,j} \\
 z^{i,j} &> 0
 \end{aligned}$$

where i , j , and t represent the warehouse, SKU, and time indices. I also have that the goods from SKU j in warehouse i at time t that are in transit are represented by $T_t^{i,j}$, and the base stock level for SKU j in warehouse i is represented by $z^{i,j}$.

7.2.5 Multi-Agent Reinforcement Learning (MARL)

Multi-Agent Reinforcement Learning (MARL) can be formulated as a Markov Game [183, 271], which is represented by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, r_i, \gamma, \mu)$. Markov games are extensions of Markov Decision Processes (MDPs) [242] that incorporate the interactions of multiple agents. In a Markov Game, \mathcal{N} is the number of agents, \mathcal{S} is the joint state space between all agents, $\mathcal{A} = \prod_{i=1}^{\mathcal{N}} \mathcal{A}^i$ is the joint action space between all agents where \mathcal{A}^i is the action space for agent i , $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the transition function, $r_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the reward function for agent i , $\gamma \in [0, 1)$ is the discount factor, and $\mu \in \Delta(\mathcal{S})$ is the initial state distribution. At each time step t , agent i takes an action $a_t^i \in \mathcal{A}^i$ from state s_t , transitions to state s_{t+1} based on the transition function \mathcal{T} , and obtains a reward $r^i(s_t, a_t)$. The goal of each agent is to learn a policy $\pi^{(i)} : \mathcal{S} \rightarrow \Delta(\mathcal{A}^i)$ that maximizes its own marginal value function $V^i(\mu) = \mathbb{E} \left[\sum_{t=0}^{H-1} \gamma^t r^i(s_t, a_t^{(i)}) \mid a_t^{(i)} \sim \pi^{(i)}(\cdot \mid s_t), s_0 \sim \mu \right]$ over a finite horizon H . The policies for each agent can be combined to form a joint policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, where $\pi(a|s) = \prod_{i=1}^{\mathcal{N}} \pi^{(i)}(a^{(i)}|s)$. This formulation means that the performance of each agent is affected not only by its own policy but the actions of all other agents.

7.2.6 Dec-POMDP for Supply Chains

A Decentralized Partially Observable Markov Decision Process (Dec-POMDP) is a generalized form of a Markov Decision Process that includes two additional components: each agent contributes to a single reward function, and the state space for each agent is dependent on an observability criterion. Thus, a Dec-POMDP can be represented by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma, \mu)$, where $\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma$, and μ are defined the same as in a Markov

Game. However, Ω_i is defined as the set of observations for agent i , and $\Omega = \Omega_1 \times \Omega_2 \dots \times \Omega_N$ is the joint observation space for all agents. $\mathcal{O} : \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow \mathbb{R}$ is the set of conditional observational probabilities that update the joint observation between all agents o , based on the next state s' selected by the transition function \mathcal{T} , and the current action a by the equation $\mathcal{O}(s', a, o) = \mathcal{P}(o \mid s', a)$. Finally, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the joint reward function for all agents. Each agent has its own policy $\pi^{(i)} : \mathcal{S} \rightarrow \Delta(\mathcal{A}^{(i)})$, which can be combined to form a joint policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ to maximize the value function $V(\mu) = \mathbb{E}[\sum_{t=0}^{H-1} \gamma^t \mathcal{R}(s_t, a_t, o_t) \mid a_t \sim \pi(\cdot \mid s_t), o_t \sim \mathcal{O}(\cdot \mid s_t, a_t), s_0 \sim \mu]$.

In the context of supply chains, a Dec-POMDP can model the decision making processes of multiple agents (SKUs) who operate under partial observability of the system. For example, the factory that makes a certain good may only observe its own inventory and the orders from the nearest distributor, and a retail store may only observe customer demand and its own stock. As a result, decisions need to be made without full information of the system, requiring each agent to rely on local observations to infer the global state. Additionally, the main goal of the supply chain is to maximize profits while minimizing operating costs, regardless of individual agent goals. This aligns with Dec-POMDPs, where the joint reward function represents a global objective over all SKUs.

7.2.7 Multi-Agent Supply Chain Inventory Management Environment

I have developed the Multi-Agent supply chain inventory management environment as a multi-echelon system of connected warehouses, inspired by [335]. This is based on the real-world process of delivering goods made from a central factory to the consumer by transport-

ing them sequentially to corresponding SKUs in warehouses until they reach the consumer. In this work, I only consider a linear sequential multi-echelon inventory management environment, which means that each warehouse has only one upstream warehouse to receive goods from, and one downstream warehouse to sell goods to.

The dynamics of the inventory management environment are as follows: first, warehouse $i \in W$ has a stock for SKU $j \in S$ at time t of $x_t^{i,j}$, and the entire stock level of the warehouse can be formulated as $x_t^i \in \mathbb{R}^n$, where $W = \{1, 2, \dots, m\}$ is the set of warehouses and $S = \{1, 2, \dots, n\}$ is the set of SKUs for each warehouse. At each time step the warehouse can order items for replenishment for each SKU $p_t^i \in \mathbb{R}^n$ from the upstream warehouse, which increases the demand $d_t^{i+1} \in \mathbb{R}^n$ for the upstream warehouse. The current warehouse also has a demand d_t^i , which can be alleviated by selling $s_t^i \in \mathbb{R}^n$ goods in the SKUs to the downstream warehouse. After a vendor lead time of $v_t^{i+1} \in \mathbb{R}^n$ for each SKU, which accounts for the transportation, loading, and unloading time of goods from the upstream warehouse to the current warehouse, $a_t^i \in \mathbb{R}^n$ goods arrive corresponding to each SKU. The current warehouse i can choose to receive a specific amount of each SKU $r_t^i \in \mathbb{R}^n$ based on the initial stock to ensure the total stock remains under the capacity of the warehouse $c_i \in \mathbb{R}^n$, and the rest $o_t^i \in \mathbb{R}^n$ goes to overflow storage. When the demand of the upstream warehouse for a particular SKU is larger than the current stock of that SKU, the current warehouse incurs a backlog of $b_t^i \in \mathbb{R}^n$. Finally, the inventory level for the current warehouse is updated to include the amount sold and the backlog. Thus, I can describe

the process mathematically as follows:

$$d_{t+1}^{i+1} = p_t^i \quad (7.1)$$

$$s_t^i = \min(d_t^i, x_t^i) \quad (7.2)$$

$$a_t^i = \sum_{k=0}^{t-1} \mathbb{I}(k + v_k^i == t) s_t^{i+1} \quad (7.3)$$

$$r_t^i = \left[a_t^i \cdot \min \left(\frac{c_i - \sum_j x_t^{i,j}}{\sum_j a_t^{i,j}}, 1 \right) \right] \quad (7.4)$$

$$x_{t+1}^i = x_t^i - s_t^i + r_t^i \quad (7.5)$$

where (1) – (5) denotes the replenishment, sale, arrival, receive, and update components for one step of the environment [335]. For each step, the action is selecting the appropriate replenishment amount for each SKU in each warehouse. The state consists of replenishment, demand, stock, arrival, vendor lead time, and other temporal data explained in the next few sections.

To calculate the reward, I must include the revenue generated from selling goods from SKUs in the current warehouse and the costs from procurement, ordering, storage, and backlog. Firstly, I have defined the total revenue gained from goods sold to the downstream warehouse as $C_r^i = \lambda_r^i s_t^i$. I have denoted the procurement cost for the current warehouse, which is the product of the unit procurement cost of the current warehouse and the amount sold by the upstream warehouse, as $C_p^i = \lambda_p^i s_t^{i+1}$. Ordering costs also have to be included which are defined as the product of the unit order cost and the total replenishment for all SKUs, denoted as $C_o^i = \lambda_o^i r_t^i$. This could include transportation and labor costs for transportation, loading, and unloading during replenishment. Additionally, there are storage costs, which could involve costs related to the maintenance of the warehouse, storage space for different SKUs, and equipment costs for

accessing and processing goods within the warehouse. This is defined as the product of the unit storage cost and the current inventory level, or $C_s = \lambda_s x_t^i$. There are also backlog costs, which arise when the demand from the order of the downstream warehouse is greater than the current inventory levels, resulting in not enough goods being transported to the downstream warehouse, which I have defined as the product of the unit backlog cost and the backlog for the current warehouse $C_b^i = \lambda_b b_t^i$. Finally, I have represented the overflow costs that arise when the replenishment amount arrives and is greater than the available space in the warehouse, resulting in a more expensive, off-warehouse storage area. I define this as the product of the unit overflow cost and the total overflow $C_f = \lambda_f o_t^i$. I define the reward (profit) function as

$$R = C_r^i - C_p^i - C_o^i - C_s^i - C_b^i = \lambda_r s_t^i - \lambda_p s_t^{i+1} - \lambda_o r_t^i - \lambda_s x_{t+1}^i - \lambda_b b_t^i$$

where $\lambda_r, \lambda_s, \lambda_p, \lambda_o, \lambda_b \in \mathbb{R}^+$.

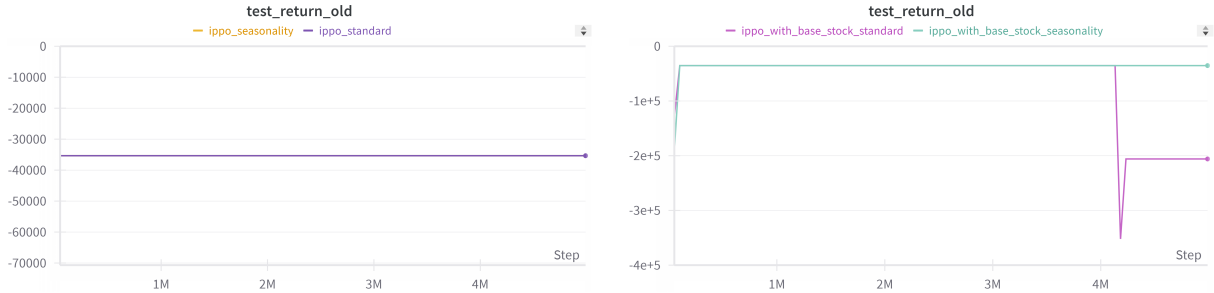


Figure 7.5: Test reward with and without seasonality data using the (a) IPPO algorithm and (b) IPPO base stock algorithm.

The profits learnt with IPPO algorithm are much more stable as shown in Figure 7.5 in comparison to the QTRAN results in Figure 7.6 for both supply chain orchestration settings, with and without seasonality information. QTRAN profits are higher than IPPO profits with more

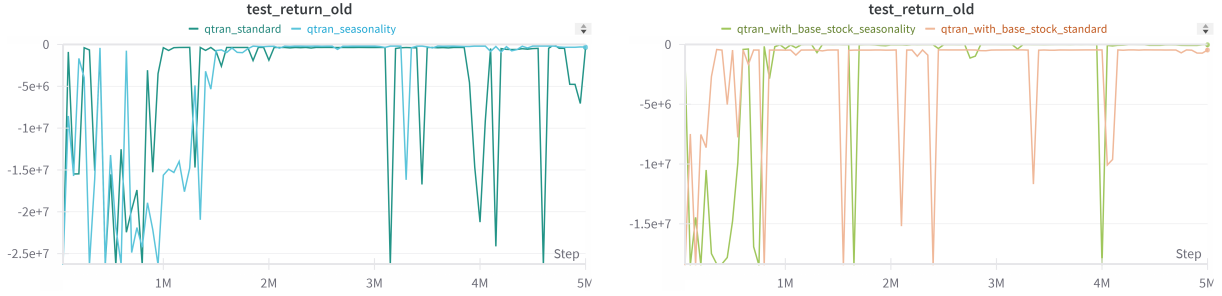


Figure 7.6: Test reward with and without seasonality data using the (a) QTRAN algorithm and (b) QTRAN base stock algorithm.

episodes, indicating the trade-off of performance vs stability. Seasonality feature in the Multi-Agent supply chain environment leads to more profits.

Seasonality

Since the state space for the multi-echelon warehouse environment contains several temporal variables, such as demand, selling price/procurement cost, and vendor lead time, I have reframed these variables to a time-series context rather than a reinforcement learning context. By treating these variables as a time series, seasonality and temporal trends can be extracted to develop more efficient and robust policies. I perform an additive seasonal decomposition, which splits the original time series data $Y[t]$ into

$$Y[t] = T[t] + S[t] + e[t]$$

where $S[t]$ is the seasonality component, $T[t]$ is the overall trend component, and $e[t]$ is the residual, or error terms. To improve the learned policies, I have embedded these features into the state space to ensure the agent learns how to exploit these temporal trends.

Outlier Economic Trends

I also investigate temporal economic trends by splitting temporal variables into different frequencies. For example, an investigation can be done on how the selling price changes per year for specific SKUs, and how the yearly trend across SKUs differs. Similarly, I have split demand, selling price, and vendor lead time into yearly, monthly, weekday/weekend, and holiday/non-holiday trends, and use Local Outlier Factor [41] to determine which SKUs have anomalous trends. Local Outlier Factor classifies an outlier with a score of -1, so to include information on the severity of the outlier and which SKU outliers are similar, I have ran DBSCAN clustering algorithm and have included the clusters in the state space.

Cold Chain SKUs

Cold chain goods in supply chains are critical because they include perishable items such as food, pharmaceuticals, and certain chemicals that must be maintained at specific temperatures throughout the supply chain process [223]. Any deviation from the required temperature range can lead to spoilage and degradation of the product. Naturally, this adds a layer of epistemic uncertainty, as unlike the aleatoric uncertainty in demand patterns, epistemic uncertainty is due to the inherent complexities and variabilities associated with perishable goods. To combat this, I can model the uncertainty of these goods and incorporate this into the framework. Nahmias [225] contends that a gamma distribution should be used, as it can accurately model the time until deterioration of perishable goods, so I sample from a gamma distribution with hyperparameters α and β . Instead of adding this information to the state space, however, I modify the selling price directly, which allows the agent to learn how to model cold chains implicitly in the policy. During

simulation, seasonal change of demand has a similar trend across seasons over a few years unlike cold chains, whose accessibility/inaccessibility determine whether the goods in the supply chain can be stored. Updates to the supply chain by the cold chain capacity can impact supply chains decisions more frequently in a temporal timespan in comparison to seasonal or outlier economic trends. In all experiments, I randomly choose 5% of the total SKUs to be cold chain SKUs, which allows me to model perishable goods from many different price ranges.

7.2.8 Generalized Multi-Agent Learning with Internal Rewards

Here, I introduced ways to improve the generalizability of each MARL algorithm.

Expectation Alignment (ELIGN)

To enhance the generalizability of each MARL algorithm, I incorporate an internal self-supervised reward model using Expectation Alignment (ELIGN) [199]. Specifically, I design a world model $f_\theta(o_i, a_i)$ that predicts the next observation o'_i . Given the partial observability between warehouses (since SKUs within a warehouse can only observe other SKUs in the same warehouse), I construct separate world models for each warehouse rather than for individual SKUs to reduce computational complexity.

Each warehouse-level world model predicts the next states for all SKUs within the warehouse. These predictions are then utilized to compute an intrinsic reward:

$$R_{in} = -\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|o'_i - f_\theta(o_i, a_i)\| \quad (7.6)$$

where $\mathcal{N}(i)$ represents the set of SKUs in the receptive field of SKU i , encompassing all SKUs

within the same warehouse.

CUriosity-based Reward from Latent Spaces (**C**URLS)

One weakness of the ELIGN reward is that it incentivizes predictability amongst agents in fully cooperative settings such as mine. For relatively simple environments such as the Multi-Particle environment [191], this is sufficient, but for my complex inventory management environment, the state space quickly scales with the number of SKUs. To address the complexity of my state space, I augment ELIGN with a cluster exploration bonus to form a novel reward I call CUriosity-based Reward from Latent Spaces (CURLS), drawing on the concept of curiosity used to describe intrinsic rewards that motivate exploration.

To construct the CURLS intrinsic reward, I begin using the ELIGN reward in Equation 7.6 as a starting point. As the agents train, I collect their observations and concatenate them into a single "state" vector. Once I have collected a sufficient number of samples, I group them into k distinct clusters using a clustering algorithm. In my experiments, I use K-means clustering, but CURLS can easily be adapted to use any clustering algorithm, such as DBSCAN [94] or OPTICS [10]. At this stage I incorporate a bonus into the intrinsic reward that boosts behavior where novel clusters are explored:

$$b(\mathbf{o}) = \begin{cases} \frac{1}{1 + n_c}, & \text{if the clustering is fitted,} \\ 0, & \text{otherwise,} \end{cases} \quad (7.7)$$

Here, \mathbf{o} is a vector composed of the observations of the agents in $\mathcal{N}(i)$ concatenated together, and n_c refers to the number of times a cluster has been "visited" over the course of the training

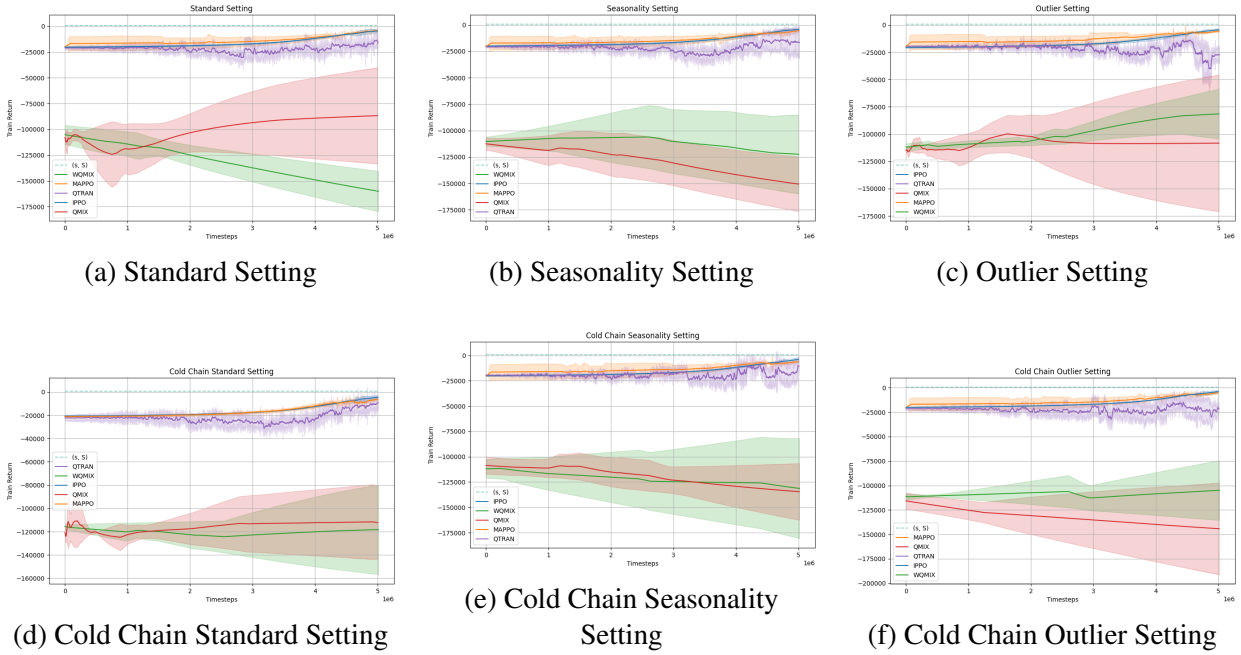


Figure 7.7: Performance comparison of reinforcement learning algorithms across different settings. (a) **Standard Setting**: Standard MARL settings as in MABIM [335]. (b) **Seasonality Setting**: Performance under settings with seasonality patterns. (c) **Outlier Setting**: Performance under settings with outlier data. (d) **Standard Setting with Cold Chain SKUs**: Standard setting with the inclusion of cold chain SKUs. (e) **Seasonality Setting with Cold Chain SKUs**: Seasonality setting with the inclusion of cold chain SKUs. (f) **Outlier Setting with Cold Chain SKUs**: Outlier setting with the inclusion of cold chain SKUs. Each subfigure shows the average performance and variability of the algorithms under the corresponding scenario.

process. Because the bonus is inversely proportional to the number of times the cluster has been visited, the agent is encouraged to be "curious" and explore unfamiliar clusters. This forms my final intrinsic reward

$$R_{final} = R_{in} + b(\mathbf{o}). \tag{7.8}$$

7.2.9 Results

This section evaluates and discusses the performance of five MARL algorithms in several supply chain settings.

Feature	STEIMAC	MABIM
Scale	✓	✓
Cooperation	✓	✓
Competition	✓	✓
Robustness	✓	✓
Generalization (Gap)	✓	✓
Seasonal Trends	✓	×
Outlier Detection	✓	×
Cold Chain SKUs	✓	×
ELIGN Generalization	✓	×
Curiosity-Driven Reward	✓	×

Table 7.3: Feature Comparison between STEIMAC and MABIM.

Experiment Details

These experiments compare the performance of 5 MARL algorithms, three off-policy algorithms: QMIX [248], QTRAN [281], WQMIX [247] and two on-policy algorithms: IPPO [85], and MAPPO [341]. Also compared with each MARL algorithm is a baseline (s, S) policy [357], which searches for optimal s and S parameters on the training set. Each algorithm is evaluated based on the mean reward over all SKUs in each warehouse, and each experiment is conducted on a single NVIDIA RTX A4000 GPU.

Seasonality and Outlier Results

The results from the standard, seasonality, and outlier settings are shown in Figures 7.7a, 7.7b, and 7.7c, respectively. Across all three settings, IPPO and MAPPO consistently achieve the highest performance and exhibit the most stable reward trajectories. QTRAN demonstrates moderate performance slightly below the on policy algorithms, but its reward trajectory is notably more unstable compared to the other algorithms. In contrast, QMIX and WQMIX maintain stable trajectories throughout learning but display the lowest performance, as they struggle to effectively

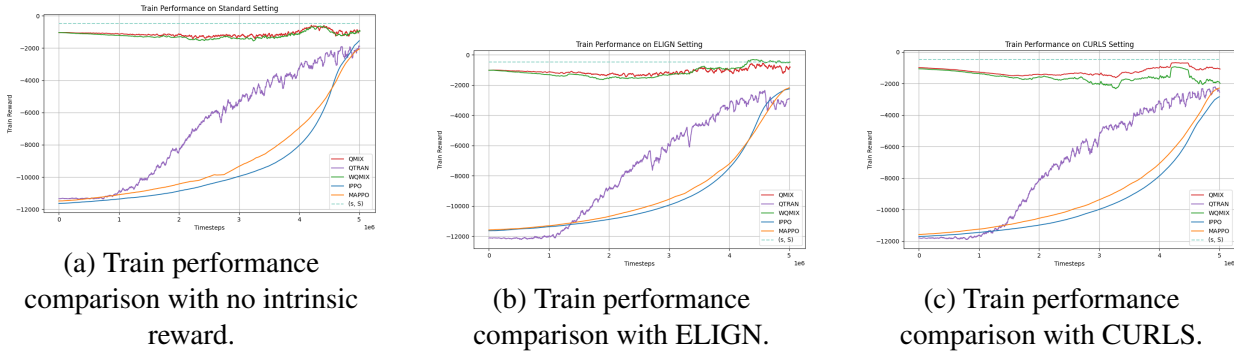


Figure 7.8: Train performance comparison of MARL algorithms with different intrinsic reward algorithms. (a) **Standard Setting**: Train Performance of Standard MARL setting as in MABIM [335]. (b) **ELIGN Setting**: Train Performance of MARL algorithms with ELIGN intrinsic rewards. (c) **CURLS Setting**: Train Performance of MARL algorithms with CURLS intrinsic rewards.

coordinate agents and leverage the available state information. In the seasonality setting, all algorithms show more stable learning trajectories compared to the outlier setting, albeit with slightly lower performance. The improved stability can be attributed to the predictable nature of seasonal patterns, which allows the algorithms to develop more consistent policies. However, introducing seasonal trends might introduce distractions in the state space, potentially causing algorithms to focus on less relevant features during policy learning. On the other hand, introducing outliers to the state space introduces more instability, but better performance, especially for the off-policy algorithms. WQMIX and QMIX have better reward trajectories than in the seasonality setting, and QTRAN maintains similar performance but has a much more variable trajectory. Finally, the on-policy algorithms see little change in the outlier setting.

Cold Chain Results

The results of the addition of cold chain SKUs to the STEIMAC framework are in Figures 7.7d, 7.7e, and 7.7f. Across all three settings (standard, seasonality, outlier), introducing cold chain SKUs has minimal impact on the on-policy algorithms, IPPO and MAPPO, as their reward

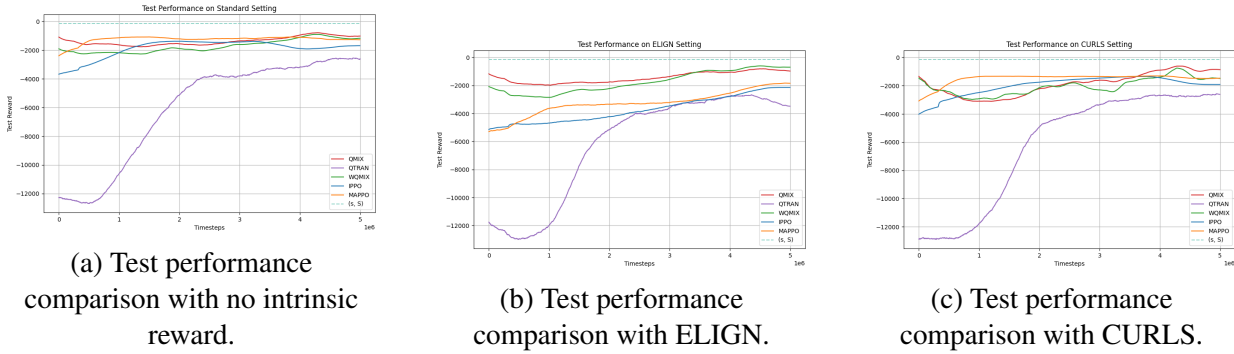


Figure 7.9: Test performance comparison of MARL algorithms with different intrinsic reward algorithms. (a) **Standard Setting**: Test Performance of Standard MARL setting as in MABIM [335]. (b) **ELIGN Setting**: Test Performance of MARL algorithms with ELIGN intrinsic rewards. (c) **CURLS Setting**: Test Performance of MARL algorithms with CURLS intrinsic rewards.

trajectories remain stable and converge to nearly breaking even, demonstrating their robustness to the uncertainty introduced by perishable goods.

However, the off-policy algorithms continue to have lower performance, with WQMIX and QMIX struggling to learn an effective policy. Interestingly, the effects of adding cold chain SKUs and introducing seasonality or outliers appear to be largely independent. The performance patterns observed when cold chains are added closely mirror those in the corresponding non-cold chain settings, suggesting that the uncertainties introduced by cold chain SKUs do not affect the performance of each algorithm when adding seasonality and outlier information to the state space.

Results with Intrinsic Rewards

Representative examples of training with different rewards are shown in Figures 7.8a, 7.8b, and 7.8c. They show typical experiments in a standard single-store setting. There are several interesting patterns. One is that during training, having an intrinsic reward tends to diverge QMIX and Weighted QMIX. The inverse of this can be seen in Figure 7.8a, where no intrinsic reward

results in the two algorithms being almost exactly the same. In contrast, WQMIX slightly outperforms QMIX with an ELIGN reward in Figure 7.8b, and QMIX outperforms WQMIX by a wider margin with a CURLS reward, as seen in Figure 7.8c. This could be due to ELIGN preferring more cooperative behavior, thus favoring WQMIX, while CURLS favors more exploratory behavior, leading to QMIX receiving higher rewards. Also, with a CURLS reward, MAPPO consistently receives a higher reward than IPPO, when with no intrinsic reward or using ELIGN, their reward was roughly on par with one another. This can be explained by MAPPO’s ability to coordinate exploration strategies between agents. This positions it to take better advantage of the cluster bonus compared to IPPO.

Similarly, some interesting trends emerge at test time, as seen in Figures 7.9a, 7.9b, and 7.9c. These observations include algorithms (besides QTRAN) improving at a roughly linear rate when using ELIGN compared to the static rewards (Figure 7.9b) when no intrinsic reward is used (Figure 7.9a) or when a CURLS intrinsic reward is used (Figure 7.9c). This is in line with the intuition that it is favorable to have cooperative agents be predictable to one another at test time to better plan the supply chain, and that it is not necessary to visit unfamiliar state spaces once training is completed.

Setting	IPPO	MAPPO	QTRAN	QMIX	WQMIX
1 Warehouse	17.99h	16.05h	17.13h	41.26h	41.0h
2 Warehouses	23.42h	20.04h	24.71h	37.26h	36.83h
3 Warehouses	33.79h	28.3h	27.36h	51.34h	51.43h

Table 7.4: Training times for each algorithm under different numbers of echelons for 200 SKUs in each warehouse.

7.2.10 Discussion

The results highlight the differences between on-policy and off-policy algorithms in handling the large and dynamic state space in the STEIMAC environment. IPPO and MAPPO, which are on-policy algorithms, consistently outperform the off-policy methods of QMIX, WQMIX, and QTRAN. This is most likely due to their ability to learn directly from the current policy's trajectory, which helps maintain stability in reward trajectories and handle the scale of 200 agents (SKUs) interacting in each of the 3 warehouses. On the other hand, the results suggest that the value-decomposition and Q-function factorization methods such as QMIX and WQMIX struggle to learn effective policies in the STEIMAC environment, even when adding outlier policies and seasonality information. This is most likely because while Q-function factorization can increase sample-efficiency, it restricts the possible generated policies. Since QMIX and WQMIX must satisfy the Individual Global Maximum (IGM) property, the constrained policies show little improvement when trying to coordinate 200 agents. This also explains the improved performance with QTRAN, as the Q-function factorization for QTRAN is more general and can handle more diverse environment conditions than QMIX and WQMIX. Another important comparison between MABIM and STEIMAC is the use of real versus synthetic data. One advantage synthetic data has over data from the real world is that real world data can be biased by factors like the sample set. Supply chain data, and especially supply chain data affected by seasonality, is influenced by the locations of the warehouses and the source of the data itself. Furthermore, the use of synthetic data, often using functions like the gamma distribution for cold chain prices, allows me to model new realistic behaviors in supply chains. If STEIMAC were to work with the real data of MABIM, I would not be able to account for seasonality and perishable cold-chain goods. This

flexibility allows me to fill the gaps left by other benchmarks without needing to painstakingly collect a similar dataset with the additional features I wish to study.

7.2.11 Conclusion and Future Work

In this study, I introduce the Seasonal Trends for Efficient Inventory Management with Multi-Agent Cooperation (STEIMAC) framework, designed to enhance inventory management strategies in multi-echelon supply chains. By incorporating temporal features such as seasonality, outliers, and cold-chain product management into the existing Multi-Agent Reinforcement Learning (MARL) benchmarks, I expand on previous benchmarks such as MABIM [335]. I provide an in-depth comparison of on-policy and off-policy MARL algorithms showing that on-policy algorithms like IPPO and MAPPO are more stable which can be suitable for small business warehouses trying to maintain stable profits in an uncertain market. Off-policy algorithms like QMIX, WQMIX and QTRAN are high risk, high reward, which do have stability challenges but learn how to get more profit over more learning steps. Additionally, to enhance the generalization and scalability of each MARL algorithm, I incorporate an intrinsic reward CURLS which is curiosity-driven, exploring latent state spaces to align agent objectives across warehouses. In future work, I aim to additional features to the inventory management environment. For example, the current approach includes a linear bus network where one warehouse only directly interacts with the previous and next warehouses in the chain. Including hierarchical tree-based networks can fully show the downstream nature of inventory management and have more diverse environment configurations. I can also have differing storage costs and capacity values for each SKU, as smaller and cheaper items such as goods can be stored in large quantities with little supervision

compared to expensive electronics that require specialized handling and security measures. Additionally, I plan to support the modeling of dwell time in warehouses leading to price stagnation or drop in commodity prices decreasing inventory profits, where supply chain agents would like to minimize dwell time. On the algorithmic side, I can also introduce communication networks between agents to enable better coordination between SKUs across warehouses. By using graph-based networks such as GNNs [285], I can scale communication as the number of warehouses increases and inventory management structure changes.

7.3 LLM Agents

After the rise of widespread LLM usage with models such as ChatGPT, interactive AI experiences have become much more commonplace. A natural extension of LLM capabilities comes in the form of allowing them to take actions, transforming them from text generation tools to Agents that have the ability to take actions of their own that affect themselves and the environment. In order to be their most trustworthy and useful, LLM Agents need to robustly determine cause and effect between actions, and they need to be able to safely and effectively work with User Agent data to provide a seamless experience while interacting with User Agents. This motivates my research on assessing the ability of LLMs to explain AI outputs and my literature review on Personalized LLM Agents.

7.3.1 Robustness of LLMs Generating AI Explanations with Zero-Shot and Few-Shot Prompting

The emergence of Large Language Models (LLMs) in the Foundation Model Landscape of Artificial Intelligence (AI) has led to its increasing usage of Generative AI applications with the aim of achieving human comprehension quality. Explainable Artificial Intelligence (XAI) gives insight on what factors impacted the AI models' outcomes by attributing them to the features in the training data. A large number of XAI models like SHAP, LIME, gradient with respect to the input, and more motivate an important question: Can a LLM agent robustly generate its preferred XAI model outputs among multiple choices that can appropriately interpret an AI model satisfying user agent preferences? This question can help reduce the complexities during data collection to train AI models and is therefore investigated with zero-shot and few-shot prompt engineering on LLMs to automatically generate which XAI model, SHAP or LIME, is preferred for the AI model, random forest classifier on two separate training datasets, evaluating the approach with human annotations. I probe the influence of prompt tokens on the LLM-Generated Explanation using XAI and remove the words in the prompt that are most important to generate the explanation in order to investigate the robustness of zero-shot and few-shot prompting for LLM-Generated Explanations.

Introduction

In recent years, Large Language Models (LLMs) have become prominent in the field of foundation models for artificial intelligence, revolutionizing natural language understanding and generation tasks, mostly for high-resource languages. These models have displayed capabili-

ties of human-like outputs in myriad applications, including text generation, machine translation [21, 24], and interactive Generative AI models like ChatGPT. However, as LLMs continue to be incorporated in more cognitive applications with good quality generations in high-resource languages, their inherent opacity has raised questions regarding their ability to explain and reason about what they output. Some previous work has shown that LLMs perform poorly on reasoning tasks [307], while other work has shown by using methods like Chain-of-Thought, which use few-shot prompting methods, LLMs can effectively reason to answer questions [326].

This Chapter evaluates the performance of two different approaches, zero-shot and few-shot prompting, to determine if LLM agents are capable of reasoning about another task: choosing between XAI outputs to explain the output of an AI model. I have a chain of instructions to give context about XAI model outputs, the AI model, the training data, predicted class label and the ground truth class label. Then I have asked the LLM agent to generate its preferred XAI model output. I evaluate the performance of the LLM on this task by comparing its labels with human annotations preferring one XAI model output over another. I investigate which set of words are important to determinate robust LLM generation by using Gradient based XAI [277] method. Removing the top four tokens from the prompt with highest feature scores impact the output preference and observing changes in generated labels helps to get insights on how effectively LLM agents can reason about XAI model outputs.

XAI Selection

My goal is to have a LLM generate its preference for one of the two XAI models, LIME or SHAP to explain Random Forest Classification. SHAP (Shapley Additive Explanations), calculates a score for each feature used by an AI model. A positive score indicates a feature contributed

more heavily to a model output while a negative score indicates a feature did not contribute as much to a model output. It calculates these weights by considering all coalitions of features in the model that can be provided to the model and then measuring the change in model output. [197]. LIME (Locally Interpretable Model-Agnostic Explanations) takes a different approach to calculate these scores by applying local perturbations to an input to a model and seeing how a model’s predictions change. By measuring the way the predictions of the model change in relation to the perturbations, LIME determines which features contribute positively or negatively to an outcome. These XAI models change their outputs depending on which AI model and dataset is used as per user requirement [260].

I generate explanations using the XAI models, LIME and SHAP, on test samples for random forest classifiers trained on two datasets. The HCMST (Couples) dataset [252] documents whether couples stay together or not and various attributes about the couple such as age, education, and income. The Diabetes dataset [5] documents whether patients have diabetes or not and other health information like insulin levels, glucose levels, and blood pressure. Figure 7.10 represents the LIME and SHAP XAI plots for a test sample explaining the goal of random forest classification on Couples dataset.

Dataset	Model	# Features	# Train	# Test
Diabetes[5]	Random Forest Classifier	18	537	231
Couples[252]	Random Forest Classifier	8	1030	442

Table 7.5: Statistics of Two Training & Testing Datasets for Random Forest Classification

LLM Prompts

Flan-T5 is a class of Large Language Model (LLM) that is instruction fine-tuned to increase performance. Flan-T5 has been shown to improve performance with instruction fine-tuning over

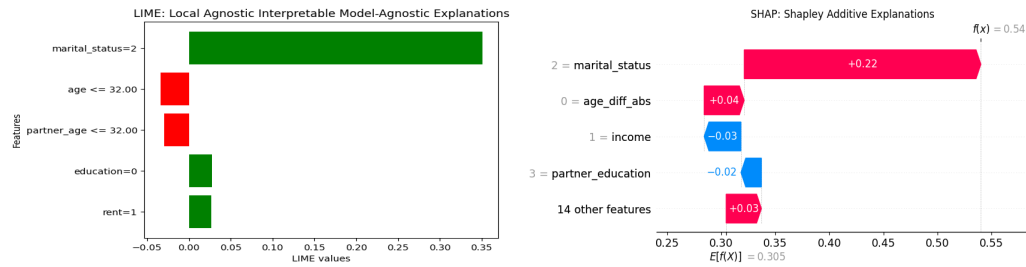


Figure 7.10: (a) LIME explanation and (b) SHAP explanation for a test sample in the Couples dataset

many different model sizes from 80M parameters to 11B and can even outperform models that have not been fine-tuned of much larger size [72]. I have given prompts to the Flan-T5-XL checkpoint to generate the LLM labels for XAI outputs.

Zero shot Zero shot prompting is a method to prompt an LLM by providing it with instructions describing the task without context [164]. In this setting, I define zero shot prompting as asking the LLM to choose between SHAP and LIME without providing it context of human annotations for a similar task.

The prompt is structured so that the LLM is given context on the dataset and model being used. In the case of Figure 7.11, it is the Couples dataset and the random forest classifier. This includes a description of the type of model being used as well as the number of training and test samples used for training and evaluating the model. After that, the features and feature values for the test sample that the XAI models were run on is given. Next, the prediction by the random forest classifier as well as the ground truth label is added to the prompt. Finally, the LIME and SHAP scores and their corresponding features and the question being asked of the LLM: to choose between the provided explanations.

```

Pick an XAI model, Shapley Additive Explanation (SHAP) or Local Interpretable Model-Agnostic Explanations (LIME), that does the best
job of explaining a prediction by an AI model.
The AI model is a random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various
sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
The training data is on couples and the classes are whether the couple stayed together, denoted by 1, or the couple broke up, denoted
by 0.
There are 1472 samples, 1030 of which have been used for training and 442 for testing the model.
The feature values for the test sample are as follows:
age=29
partner_age=29
age_diff_abs=0
children=0.0
visits_relatives=4
education=bachelor's degree or higher
marital_status=married
partner_education=some college
gender=female
house=one-family house detached from any other house
income=$100,000 to $124,999
msa=metro
rent=owned or being bought by you or someone in your household
political=democrat
religion=catholic
workworking = self-employed
gender_older=0
education_difference=1
The prediction for the test sample using Random Forest Classifier is 1.0
The ground truth label for the test sample is 1.0
LIME, which means Local Interpretable Model-Agnostic Explanations, has scores for the features marital_status, age, house,
partner_age, partner_education, as 0.3261, -0.0547, 0.051, -0.0462, -0.0294.
SHAP, which means Shapley Additive Explanation, has scores for the features marital_status, income, partner_education, age,
partner_age, as 0.153, -0.0478, -0.0436, -0.0249, -0.0245.
Keep in mind that LIME and SHAP are very comparable models.
Using only the information just provided, please make the following choice:
1.) The Local interpretable Model-Agnostic Explanations XAI model's output is preferred
2.) The Shapley Additive Explanations XAI model's output is preferred

```

Figure 7.11: An example of a zero-shot prompt for Flan-T5-XL

Few shot Few shot prompting can be contrasted with zero-shot prompting in that some context is provided to the LLM of the task it must do in addition to instructions to complete the task [164]. I provide this context with one example each of both SHAP and LIME human annotations along with the SHAP and LIME feature scores that were given to the human annotator, as seen in. Additional specificity is also provided in the query to the LLM by asking it to choose the XAI model whose outputs are most similar to the ones provided in the few shot context, as seen in Figure 7.12.

Results

I evaluate performance on the labelling task by comparing LLM labels with that of a human annotator. I randomly sampled 200 test samples, 100 from the Couples dataset and 100 from the Diabetes dataset, based on which I construct the prompts for the zero shot and few shot setting. A human annotator was provided the same prompts given to the LLM agent and asked to choose between SHAP and LIME. Their annotations were validated by another human annotator agent.

Pick an XAI model, Shapley Additive Explanation (SHAP) or Local Interpretable Model-Agnostic Explanations (LIME), that does the best job of explaining a prediction by an AI model. The AI model is a random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

The training data is on couples and the classes are whether the couple stayed together, denoted by 1, or the couple broke up, denoted by 0. There are 1472 samples, 1030 of which have been used for training and 442 for testing the model.

The features of this dataset are as follows: age, partner_age, age_diff_obs, children, visits_relatives, education, marital_status, partner_education, education, marital_status, partner_education, gender, house, income, msa, rent, political, religion, work, gender_older, and education_difference.

To give some context for this task, this is an example of a choice made by a human on a different test sample:

When Local Interpretable Model-Agnostic Explanations had scores for the features marital_status, partner_age, education_difference, age, partner_education, as 0.3477, 0.0552, 0.033, -0.026 and Shapley Additive Explanations had scores for the features marital_status, age, partner_age, house, work, as 0.1311, 0.0447, 0.0291, 0.0168, 0.0168, the human chose Local Interpretable Model-Agnostic Explanations (LIME) as having a preferable output.

Another example of a choice made by a human on a different test sample:

When Local Interpretable Model-Agnostic Explanations has scores for the features marital_status, partner_education, msa, house, visits_relatives, as -0.3452, 0.0408, -0.0203, 0.0158, -0.0122 and Shapley Additive Explanations had scores for the features marital_status, partner_education, education_difference, gender_older, partner_age, as -0.3333, 0.0248, -0.0217, 0.0192, 0.0168, the human chose Shapley Additive Explanations (SHAP) as having a preferable output.

The feature values for the test sample you will be asked to classify are as follows:

```

age=29
partner_age=29
age_diff_obs=0
children=0
visits_relatives=4
education=bachelor's degree or higher
marital_status=married
partner_education=some college
gender=female
house=one-family house detached from any other house
income=$100,000 to $124,999
msa=metro
rent=owned or being bought by you or someone in your household
political=democrat
religion=catholic
work=working - self-employed
gender_older=0
education_difference=1

```

The prediction for the test sample you will be asked to classify using Random Forest Classifier is 1.0
The ground truth label for the test sample you will be asked to classify is 1.0
LIME, which means Local Interpretable Model-Agnostic Explanations, has scores for the features marital_status, age, house, partner_age, partner_education, are, respectively, 0.3261, -0.0547, 0.051, -0.0462, -0.0294.
SHAP, which means Shapley Additive Explanation, has scores for the features marital_status, income, partner_education, age, partner_age, as 0.153, -0.0478, -0.0436, -0.0249, -0.0245.
Choose the option that has scores most similar to the examples you've already seen.
1.) The SHAP XAI model's output is preferred
2.) The LIME XAI model's output is preferred

Figure 7.12: An example of a few-shot prompt for Flan-T5-XL

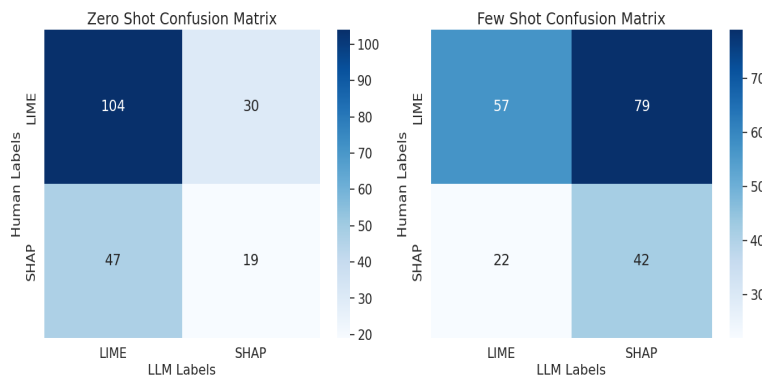


Figure 7.13: Confusion Matrices for the Couples dataset comparing zero shot and few shot prompting

Dataset	Number of Samples	Prompting Strategy	Accuracy
Diabetes	200	Zero Shot	54%
Couples	200	Zero Shot	61.5%
Diabetes	200	Few Shot	44.5%
Couples	200	Few Shot	49.5%

Table 7.6: Accuracy of LLM labelling with different prompting strategies

Analysis A decrease in performance is observed for LLM-Generated automatic explanation labelling with the few-shot prompting strategy, for both the Couples and Diabetes datasets. The performance on the Couples dataset drops 12% while the performance on the Diabetes dataset drops 9.5%, as seen in Table 7.6 and Figure 7.13. This shows the increased context length does not necessarily help the LLM gain understanding of how a task should be performed. Due to the $O(n^2)$ computational complexity of self-attention [309] used in transformers, LLMs can get exponentially worse at attending to larger contexts. This can provide an explanation for why adding more context hurts performance more than it helps it.

Robust Interpretation of LLM-Generated Explanations

To determine the important words in zero-shot and few-shot prompting that decide the robust generation by LLMs, I use a XAI model, computing gradients of outputs with input features [277] method to assign scores for each of the tokens generated by the model’s tokenizer, as seen in Figure 7.14. The higher these scores are, the more the model rely on them to generate its output. I then test robustness by removing the top four highest scored tokens. I have observed that out of the ten randomly sampled prompts, the LLM’s generated XAI model output preference flipped for seven of the prompts with respect to their previous generation. This gives an insight on the set of important words that ensure the robustness of LLM generation.

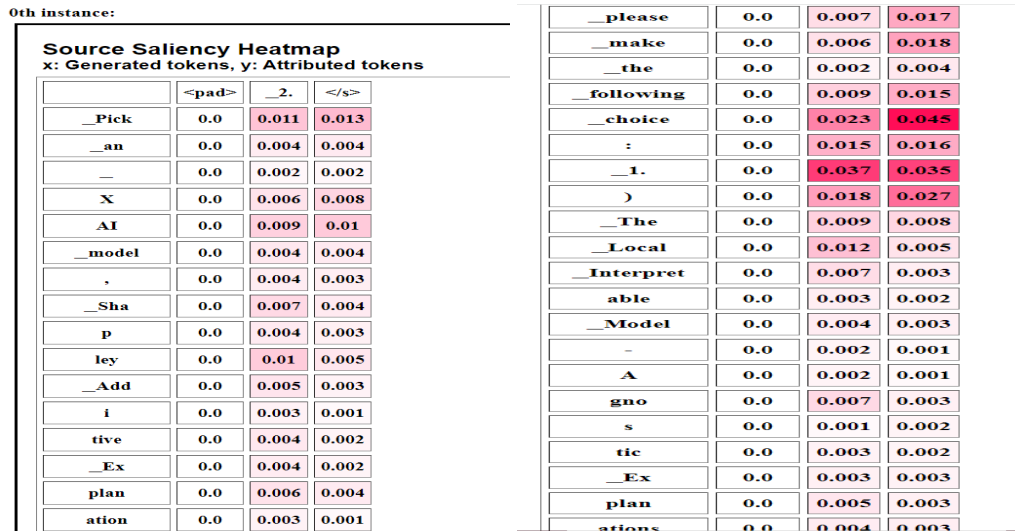


Figure 7.14: The rows of a table of the most important tokens in the prompt impacting the generation as scored by the Input X Gradient method [277]

7.3.2 Personalizable LLM Agents

Robustness for explainable AI methods on LLMs directly relates to creating personalized LLM Agents, as interpretable LLMs are a prerequisite for a trustworthy LLM Agent that can access user data to provide a more personalized experience. This section is a literature review of LLM Agent and User Agent interaction to provide personalized LLM Agents.

Several studies incorporate personalized generation for a specific domain into generative models by including domain-specific information or experience. For example, [177] incorporates user-item links to knowledge graphs to develop a heterogeneous knowledge graph, which is embedded in a text generation model, such as an RNN. Additional studies have used Generative Adversarial Networks (GANs) for personalized text generation [123, 343] by guiding the generation process with additional context [179, 227] or structured knowledge [289]. However, while these studies do add additional context, they do not use this information fully to output personalized text specific to the preferences of the user.

As a result, studies have shifted towards using Large Language Models (LLMs) for generating personalized text. For example, given an immediate context and past context for a specific user over several documents, [176] develop a personalized text generation model by feeding top-ranked summarized snippets of all the documents into an LLM. Another study by [178] develops a personalized text generation and recommendation model by giving a list of previous purchases by the user as input and using beam search to generate personalized search queries for a recommendation model.

Multi-Agent communication Many studies have used Multi-Agent Reinforcement Learning (MARL) to handle effective communication between agents in a common environment, such as in sensor networks [332], robotics [116], and games [172]. For example, [139] develop a Multi-Agent communication system by constraining it to an LLM prior as a reference policy using a regularization term. However, this approach requires human interaction from the LLM to ensure that communication between each agent does not deviate from the prior. Additionally, [126] addresses scalability in Multi-Agent communication by ensuring agents adopt a local communication strategy that sends messages only to agents that can be observed without modeling all agents.

Several works in Multi-Agent communication involve using Graph Neural Networks (GNNs), as GNNs can store communication between agent i and agent j by the edge that connects node i to node j [136, 288]. For example, [221] developed a variant of Graph Neural Networks called Graph Decision Networks designed to handle Multi-Agent communication. This is done by passing a graph containing observation, environment, and communication information between agents into a GNN, which outputs actions for each agent.

Personalizable Representation Learning in Spatio-Temporal and Low-Resource Contexts

A significant rise in the applications of Artificial Intelligence (AI) models has led to the necessity of underlying representations to be more personalizable with diverse user agents of AI models. Cognitive AI models like Large Language Models (LLMs), [325], Visual-Language Models (VLMs) [349] or decision making AI models like decision transformers, [63], robotic transformers [43] among other models have not been explainable or interpretable in contextual ways to multiple users. Multilingual LLMs can ensure economic benefits of AI models to reach millions of people speaking low-resource languages like Swahili, Yoruba in Africa; Nepali, Pashto in Asia, Native American languages etc. LLMs in high resource languages have significant gaps in reasoning, hallucinations, planning, memorization or harmful text generation among other applications [151].

The representations for these AI models from underlying structures like Part-of-Speech (PoS) tags [14, 91] can be used to learn about prompt syntax in LLMs. It also helps to learn Multilingual LLMs for low resource languages like Swahili better aligned with high resource pivot languages having PoS tags. E.g. a Multilingual LLM generating text in low-resource African languages like Setswana can improve in textual generation with corresponding translated prompts in English along with their underlying PoS tags.

The prompts can be labeled with dialog act types like identifying statements and questions. These feature representations help in personalizing [258] the output of AI models like LLMs both for low-resource [15] and also high-resource languages along with giving insights for lack of capabilities like repeated generation due to underlying syntactic structure in LLM prompts.

Representations for these AI models can have generic features tagging persons,[100], or-

ganizations, locations [282] or time [8] giving additional contexts for sparse settings. Context switching can be documented with bit-level features monitoring changes in these feature tags along a prompt which helps the AI model's outputs to be personalizable to the user agent's changing requirements. Temporal context helps to improve on factual accuracy of the AI model's output.

Personalizing representation learning opens a new paradigm of fairness to make AI models more accessible to users based on their context and needs.

7.3.3 Conclusion

LLMs like Flan-T5-XL can be used to generate AI explanations automatically for AI models like Random Forest Classification with corresponding training data. Corresponding to manually annotated preferences of SHAP or LIME XAI models, experiments lead to a 61.5% accuracy and 54% accuracy for the Couples and the Diabetes datasets with zero-shot prompting respectively and 49.5% accuracy and 44.5% accuracy for the above mentioned datasets respectively with few-shot prompting. For the human annotator agent's labels on randomly sampled data items, zero-shot prompting is closer to human preferences vis-à-vis few-shot prompting. The important words in the prompt impacting the generation of explanation models are identified using an XAI model computing gradients of output scores with respect to input prompt. When these important words are removed, the explanation preference generated by the LLM flips for most of the prompts giving insights into robustness. XAI models provides an understanding of which important words in the prompt control the robustness of LLM generated AI Explanations.

8.1 Goal-Conditioned Reinforcement Learning to Recommend AI Explanations

The large-scale usage of Artificial Intelligence (AI) models has made it important to explain their outputs subject to requirements and goals for using these models. The definition of goals in Goal-conditioned Reinforcement Learning (GCRL) aligns with the task of recommending an appropriate explanation among Explainable AI (XAI) models like SHAP or LIME that is most interpretive for specific AI models. The initial focus has been on two goals of training random forest classifier to classify different training data in order to find appropriate explanations. SlateQ recommendation system is used for simulation where the underlying RecSim environment has a slate of documents with different quantity scores representing different goals.

8.1.1 Introduction

In the realm of artificial intelligence, the pursuit of Explainable AI (XAI) models has never been more crucial for transparency. Understanding the inner workings of AI models has become imperative to ensure their reliability, fairness, and accountability. XAI tools have emerged to provide insights into AI model behavior and decisions. There exist a variety of XAI models that

aim to provide insight into the behavior of complex AI models, which presents a challenge: how can the most appropriate explanations be effectively recommended that elucidates the behavior of AI models as per user goals?

This open research challenge is explored by connecting Goal-Conditioned Reinforcement Learning (GCRL) to a recommender system that can select relevant explanations. The recommender system's objective is to learn a policy that selects an XAI model's outputs most aligned to user goals. Goals are the specific task done by the AI model.

It can be challenging to represent goals in recommender systems [66, 318]. There can be unified goals while selecting a set of items such as selecting different XAI model outputs for a meta-task like agriculture planning using precipitation predictions from an AI model. Likewise, there can be individual goals in selecting different XAI model outputs. Different tasks like disaster recovery or irrigation planning need different characteristics of XAI model outputs based on an interpretation of the individual preferences of the users in different scenarios. This makes the GCRL based recommender system pertinent to recommend the appropriate XAI model output for different tasks.

The SlateQ [144] recommender system is used to simulate this in the RecSim environment [143], recommending from a slate of different documents, which are selected by their relevance to the user. This research has been the motivation to recommend slates of XAI explanations elucidating AI model outputs. These slates are constructed considering user preferences and generated labels from LLMs preferring explanations generated from XAI models.

Shapley Additive Explanations (SHAP), calculates a score for each feature in the AI model, which represents its weight to the model output. It does this by approximating the effect of removing a feature from the model and then doing this for all subsets of features in the model.

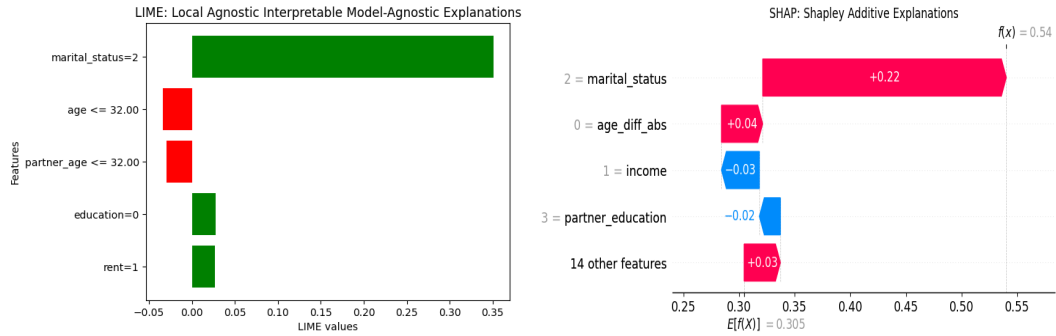


Figure 8.1: (a) LIME explanation and (b) SHAP explanation for a test sample in the couples dataset

[197]. LIME (Locally Interpretable Model-Agnostic Explanations) calculates scores for each feature in the model by measuring model behavior in response to applying small perturbations to an input, thereby generating local explanations for test samples [250]. It is important to decide when to use SHAP or LIME depending on the goal of which machine learning model is to be explained [260].

8.1.2 Offline Explanations Dataset

Explanations are generated using the XAI models LIME and SHAP on the corresponding test samples for random forest classifiers trained on the Diabetes dataset [5] and the HCMST dataset [252] as different goals. Figure 8.1 represents the LIME and SHAP XAI plots for a test sample explaining the goal of random forest classification on couples dataset.

Goal	# Train	# Test
Explaining Random Forest Classifier trained on Diabetes dataset [5]	537	231
Explaining Random Forest Classifier trained on Couples dataset [252]	1030	442

Table 8.1: Statistics of Training & Testing Datasets for two AI model goals

8.1.3 Goal-Conditioned Recommendations

State representations in SlateQ recommender system (RecSys) are static user features and historical user preference. These user features are attributes of the LIME and SHAP XAI models like latency, training data sparsity, consistency and other semantic features, which vary across different test samples. The current state is updated when an item like an XAI model output is selected by the user from the slate of explanations, updating the known user preferences for the aforementioned features. The user selects item i from the slate A with unnormalized probability $v(s, i)$ where v is some function [144] which leads to a transition to the next state in the trajectory of the user agent. The components for a goal-conditioned Markov Decision Process (MDP) in the RecSys setup for explanations are summarized in Table 8.2 with simulation results in Figure 8.2.

Let any XAI model output be θ_{x_i} for the AI model output x_i , or goal. Let the user feature values

	Descriptions
States	User preference for explanations
Actions	User choice of XAI model explanation
Transitions	Deterministic updates to the known user preferences for explanations
Reward	Assigned when the model selects an explanation that the user also prefers
Goals	Recommending XAI model outputs appropriate for specific AI models

Table 8.2: Summary of Goal-conditioned Reinforcement Learning RecSys for Explanations

be represented as $f_u^1, f_u^2, \dots \in F_u$ where F_u is the set of all feature values for the user. Let the XAI output features be represented as $f_{\theta_{x_i}}^1, f_{\theta_{x_i}}^2, \dots \in F_{\theta_{x_i}}$. Then $\theta_{x_i}^*$ is defined as the XAI model output minimizing the distance between F_u and $F_{\theta_{x_i}}$ among a slate of other XAI model outputs for x_i . The reward can be defined as the influence of user engagement in selecting recommended explanations from a particular slate that is aligned with their goals.

The actions for the recommender system are the choice of XAI model output to present

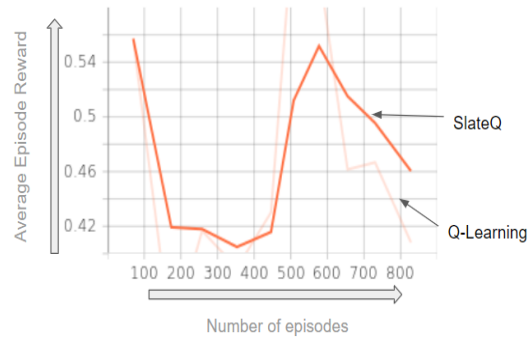


Figure 8.2: Average Episode Reward across time-steps for two goals while simulating with SlateQ RecSys in comparison with a Q-learning approach

to the user. Each XAI model output will have different values for the aforementioned features, and the recommender system will attempt to select an explanation that has feature values most closely matching those of the user's preference, θ_x^* . Necessarily, there will be one or more XAI model outputs for the same AI output as the chosen XAI model output θ'_x . The slate that is presented to the user will then consist of the recommender system selected XAI output as well as the additional non-selected XAI outputs for that AI model output. Transitions in the model are the user choice of explanations from the slate. This will update the known user preferences for features, or state. Transitions are deterministic. Reward is assigned to the model when the user selects the same XAI model output from the slate that the model itself chose.

Conclusion and Future Work

Future work involves AI explanation goal representations aligned with requirements of user agents in multiple scenarios which can be validated with human feedback along with the development of benchmarks to evaluate the quality of goals.

8.2 Multi-Agent Recommender System for Explainable AI

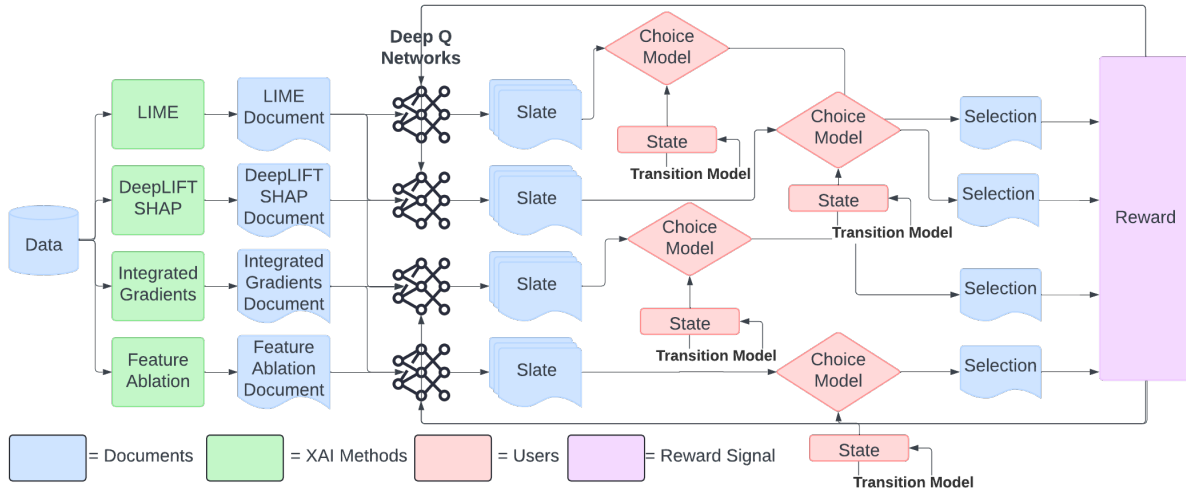


Figure 8.3: Multi-Agent XAI Recommender System Architecture

The deployment of Artificial Intelligence (AI) models for real world applications has led to a demand for explaining the outputs of AI models while considering different goals of AI users, accomplished through eXplainable AI (XAI). It is important as there are different XAI methods which can serve many aspects of XAI users' needs for fast, robust, accurate, or consistent explanations of the AI model. This motivates my research work on a Multi-Agent Recommender System (RecSys) to recommend AI explanations that can serve the goals of different user agents and are relevant to the tasks done by the AI model. I have experimented with several XAI metrics like latency and robustness scores to simulate and understand the features that influence user agents' to select an XAI method from a slate of recommended XAI methods, ordered based on their higher priority of recommendation. My research is the first work recommending four XAI methods, DeepLIFT SHAP, LIME, Integrated Gradients and Feature Ablation to multiple user agents' based on their goals and requirements to understand the AI model's tasks. I show

that when user agents' having similar interests communicate their goals, it can lead to an improvement in the diversity of XAI method recommendations. My Multi-Agent RecSys has been developed with 4 users, having common and diverging goals, having the capability to implicitly communicate their goals along the RecSys shaping the reward that scores the alignment of users' goals with XAI methods' performance across different tabular Machine Learning tasks like regression on California Housing prices or classification on diabetes datasets. I have done comparative analysis with Collaborative Interactive Recommender (CIR) systems using Deep Q Learning, and Random Strategies. I find that the Deep Multi-Agent slate-Q approach of XAI RecSys maximizes exploration of XAI methods in collaboration with the goals of user agents' while improving the user engagement and diversity of recommendations in simulation.

8.2.1 Introduction

Artificial Intelligence (AI) is increasingly being applied to real-world problems, especially in domains that involve complex socio-economic factors. One area where Artificial Intelligence can be applied is the estimation of housing prices. This is a task that influences the decision-making processes of buyers, sellers, realtors, and financial institutions. AI models can take into account many different variables such as median income of the neighborhood, location, and number of bedrooms to give an accurate estimate for the price of a house. The outputs of these AI models can have far-reaching consequences, from influencing government policy for addressing homelessness to informing corporate strategies for employee accommodation to assessing local purchasing power.

The demand of using AI models for improving decision making in such real world problems

has led the emergence of different Explainable AI (XAI) methods such as DeepLIFT SHAP [197], LIME [250], Integrated Gradients [292] and Feature Ablation [130] which can give insights into the influence of features on the decision making capability of an AI model.

This proliferation of XAI Methods opens up the problem of recommending XAI methods that align with user goals, satisfying requirements for different AI tasks like regression problems on California housing prices or predicting the progression of diabetes. In these problems, the user may prefer faster XAI methods, robust XAI Methods, or some option in the middle. Some objectives may matter more to the user, like explaining the AI model faster to make a time-sensitive decision. Users also have common goals shared explicitly based on mutual interests. For example, consider scenario where a buyer agent wants to buy a house. A banker agent may want to give a loan to the buyer for that purchase only after making sure that the loan can be repaid based on the housing prices and economic factors.

The advent of Goal Conditioned Recommender Systems [16] provides a framework to model user goals according to which items can be recommended. Such Recommender Systems (RecSys) are categorized with Collaborative Interactive Recommender Systems (CIR) [143] which allow a context window offered by a sequence of interactions between the user and the RecSys to explore good items to be recommended or to maximize the engagement of the user agents in the short term and the long term. This exploration can be achieved with Reinforcement Learning (RL) which inspires my Multi-Agent Reinforcement Learning (MARL) framework implicitly communicating among user agents by using their proportional goals based on mutual interests to shape their joint rewards. In my case, the item is the XAI output which is recommended by my CIR. I explore and recommend a slate of XAI methods to promote diverse recommendations, out of which my user choice model selects an XAI output. This is then used to calculate a recommen-

dition response quality score as the reward measuring how aligned the attributes of the ML use case were to the goals of the user agents. This quality score is used to compute the joint reward for all agents shaped by their corresponding goals for the specific use case. Thereafter, I take the preference of the user agents for next ML use cases while updating it with an User Transition Model that updates the user agent's preference for the next ML use case with the experience of the user's preferences from previous use cases, based on an idea called selfplay. The idea of introspection arising from the idea of buyer's remorse in psychology inspires me [253] to model the recommendation problem as a selfplay problem from MARL, with the users interacting with past iterations of the choices they made to select a better recommendation.

I have developed my Multi-Agent XAI Recommender System, illustrated in Figure 8.3, by training Multi-Layer Perceptron (MLP) models with four layers to do the regression task on the California Housing dataset and the regression task on the Diabetes dataset. For each of the XAI features I consider (latency, infidelity, and sensitivity), I simulate the user preference for that feature on a scale from 1 to 5, 1 meaning the user cares very little about that feature and 5 meaning they care a great deal about it. My recommender system composes a slate of XAI outputs for a data item from either of the two datasets. Each user has a Q-Network that is used to explore the expected alignment with the user of each possible slate. For one possible use case let's assume the user is recommended a DeepLIFT SHAP item and an Integrated Gradients item explaining the output from a model taking a California Housing example as input. The user chooses one of these two items using a Multinomial Proportional choice distribution roughly based on how similar the user preferences are to the features of the XAI item relative to other items. This alignment is used as input to the reward function, which provides feedback to the recommender Q-Network so the recommended slates will align with the users more next time. These rewards

are useful to compute a joint network either by summation with Value Decomposition Network (VDN) [294] or by factorizing Q functions with QMIX [248] consisting of an Agent Network for a Q value per agent and a mixing network to calculate the joint Q values, as seen in the right of Figure 8.3.

I apply my Multi-Agent Recommender System for the specific application of recommending Explainable AI (XAI) outputs. XAI methods are often used post-hoc to interpret the impact of its input features on its outcomes. A large number of XAI methods to explain AI models for tasks like classification or generation lead to the important research question of which XAI method should be used to satisfy the user's goals. A Multi-Agent Recommender System allows users to adapt their decisions based on past actions. For example, the DeepLIFT SHAP XAI method may have been used to explain the impact of features on the output classification but it takes a long time to generate its outputs which may not be ideal for explaining time-sensitive classification of economic products. So, a LIME based XAI method may be useful in this case as LIME runs faster than DeepLIFT SHAP over classification outcomes.

My contributions are as follows:

1. I present a novel recommender system capable of determining which explainable AI outputs should be recommended to a user agent. This allows for the XAI outputs to be tailored to user goals relevant to the tasks performed by the AI model.
2. This work compares four popular XAI methods: DeepLIFT SHAP, LIME, Integrated Gradients, and Feature Ablation in a multi-user environment. This comparison is based on the user goals and requirements in understanding the AI model's output.
3. I show that the recommender system is capable of recommending all four XAI methods de-

pending on user goals, showcasing the advantages and disadvantages of using each method.

4. My comparative analysis of collaborative Q-learning reward functions compared to non-agentic random baselines shows an increased ability for user agents with disparate goals to implicitly negotiate.

8.2.2 Use Cases of Multi-Agent XAI RecSys

A recommender system for personalized XAI outputs that satisfies the requirements of user goals has several use cases. One use case is finding the most appropriate XAI feature attribution method for explaining the output of a model for predicting house prices, satisfying the requirements of different users like home buyers, sellers, realtors and bankers who have their own goals. A home buyer with limited budget may care about robustly understanding learning of predicted housing prices that can be influenced by house condition, location, and economic factors to save budget with their house purchase. While a home buyer with no budget worry but has to quickly move to a new city because of a job will need faster explanations to make a quicker decision. A home seller may want an XAI attribution method to justify specific features of the home that justify the housing price. A Realtor can introduce a buyer to a seller and a buyer to banks for mortgages with the incentive of getting a commission, so they would need to know which underlying features can help match suitable buyers to sellers. A banker can need robust and fast explanations to understand the factors like location and economics influencing the housing price to make a decision of granting loans, ascertaining that they'll be getting repaid. All these users may have mutual interest during their interactions that guides them to tune their initial goals according to other user agents.

Another use case is the healthcare system, specifically in the diagnosis and prevention of diabetes. In this situation, one user may be a patient at a hospital being screened for diabetes who wants to understand the underlying parameters that can increase their risk to diabetes which can help their medication or make lifestyle decisions faster for mitigating diabetes. Another user may be a doctor responsible for curing the patient and wants fast and robust explanations in order to quickly and correctly help patients suffering from severe diabetes. Other agents like a pharmaceutical representative wants to anticipate which features impact diabetes to accordingly reflect the patient symptoms to the company and sell medicines convincing the doctor that their medicines will not have side effects for diabetes prediction features like blood pressure. The health insurance agent has an interest in such explanations to ensure healthcare coverage is efficiently given to their patient consumers and their claims payment can be optimal.

I simulate these AI use cases based on real-world California housing prices regression and diabetes prediction tasks to show that my XAI RecSys can serve generally XAI users with different goals involved in these situations. I describe the datasets further in Table 8.3. The diversity of user’s preferences and goals while having mutual interests motivating users to avoid their goals has been considered in order to serve more robust XAI recommendations for all user agents.

Dataset	# Data Points
California Housing [233]	20640
Diabetes [92]	442

Table 8.3: Statistics of Training & Testing Datasets for two AI Models

8.2.3 Related Works

Collaborative Interactive Recommendations

The objective of recommender systems include not just short term satisfaction to maximize user engagement but also improve it in the long term and make sure recommendations address user goals. Collaborative Interactive Recommenders (CIRs) address this where the Recommender Agent explores the set of options that satisfies user goals' in the short-term and in the long-term in collaboration with the users, making use of interactive user feedback. This differs from Conversational Recommender Systems (CRSs), which more directly solicit feedback from the user through Natural Language conversations. CRSs may provide an intuitive interface, but they can be less responsive to evolving user preferences, which is a major factor to consider in use cases where user agents, each with their own goals and interests, must collaborate with other agents.

Existing libraries that simulate users of CIRs [143] provide a configurable platform for authoring simulation environments for recommender systems that naturally supports sequential interaction with users. The recommendation options are explored with reinforcement learning based user modeling environments based on which new environments including my research can be supported for sequential interactive recommendation problems.

Other frameworks like [348] simulate user-item interactions in recommender systems through agent-based collaborative filtering in which both users and items are considered as agents. These agents can demonstrate personalized behaviors similar to those of real-world individuals, spearheading the development of next-generation user behavior simulation. However

they do not consider long-term user engagement along with short-term user engagement like RecSim does.

To simulate closer to the real world, session-based recommender systems like [127] models recommendation items based only on interactions in the current session for anonymous users, while providing a simulator and a recommender model. However, they have not also addressed user engagement and satisfaction of goals of single users like RecSim has done.

Some libraries such as [344] make use of findings that Reinforcement-Learning (RL) based Recommender Systems (RS) enhance long-term user engagement. The EasyRL4Rec code library, designed specifically for RL-based RSs, provides lightweight and diverse RL environments based on five public datasets and includes core modules with rich options. This library supported many RL algorithms but few ML use cases. Since my research focuses on investigating XAI results on some ML use cases and demonstrate diverse, robust and aligned recommendations satisfying the goals of different users for the first time, I plan to extend my XAI RecSys to satisfy these RL algorithms in future.

Some methods try to gain more knowledge about the user through cookies [210]. Cookies enable more accurate identification and tracking of user behaviour in a recommender systems. The impact of user's consent decision through cookies on the recommender system's ability to learn about user engagement data has been investigated in the context of an engagement-driven recommender system using simulation.

More works such as [243] address the issue of evaluation of interactive recommender systems by simulation of user interactions with the system as a part of the evaluation process.

Works such as [274] the user behaviour in recommender systems inspired by click models for Web search. The proposed models outperform baselines on the ContentWise and RL4RS

datasets and can be used in RS simulators. More data-driven approaches to RL simulation such as SARDINE, a flexible and interpretable recommendation simulator, have been proposed to accelerate research in interactive and data-driven recommender systems. LLMs can also be used for Recommender System simulation, such as in Agent4Rec, a user simulator in recommendation, which leverages LLM-empowered generative agents equipped with user profile, memory, and actions modules that are specifically tailored for the recommender system.

8.2.4 *Review of Multi-Agent Reinforcement Learning*

More examples of the advantages of self-play can be found in [77]. In a Multi-Agent setting, the other agents can learn. This makes the environment non-stationary, and emphasizes the importance of learning for each agent. Two minimal properties of a good Multi-Agent reinforcement learning algorithm are the ability to learn the optimal policy against an opponent that is stationary or may become stationary, and convergence to the Nash Equilibrium if all agents have the same policy. To achieve these goals, the authors propose AWESOME: an algorithm that has agents adapt their strategies when their opponents appear stationary, and retreat to an equilibrium when opponents are adapting.

The competition amongst agents fostered by techniques such as self-play can result in interesting emergent behavior, as in [26]. In this work the authors observe that when multiple agents trained using self-play are put into simple environments, it is possible for complex behaviors to be learned. In particular, self-play sees great results in competitive environments. Rather than constantly playing against the most recent version of an opponent, select a historical version of the opponent at random so one agent does not run away with the lead and dominate everyone else. Different policies can also be selected at random to avoid overfitting.

Chen et al. [66] survey serves to give me a review of the state-of-the-art in recommender systems that employ deep learning, including a problem statement for the task of making recommendations to a user, as well as overviews of sophisticated strategies for the different tasks encompassed by this paradigm using Multi-Agent Reinforcement Learning which is capable of handling multiple tasks. Feng et al. [99] addresses two weaknesses that happen when scenarios are optimized individually: there is no collaboration during the optimization and there is no way to model correlation between the optimizations. To this end, the authors present Multi-Agent Recurrent Deterministic Policy Gradient (MA-RDPG). The work makes the point that even in non-cooperative scenarios, collaboration between agents is necessary to maximize total return. Carroll et al. [54] studies how to predict the affect of different policies for Reinforcement-Learning based Recommender Systems on their users' preferences. To observe how a new policy influence its users, it is necessary to have access to a set of historical data on how the users interact with the old policy.

Multi-Agent Reinforcement Learning

Reinforcement Learning (RL) is an useful learning framework to help explore recommendation choices by CIRs. Multi-Agent Reinforcement Learning is an RL paradigm capable of modeling systems where many agents can interact to solve problems such as recommender systems which does many complicated tasks, many of which can be modeled to be done by agents. Marivate et al. [204] applies Multi-Agent recommender systems to the task of recommending training courses for engineers. At a high level, the agents each have simple tasks to solve, and combine their efforts to produce high-quality recommendations. One agent, the recommender, is responsible for ordering courses based on a search query. The other agent is the information re-

trieval agent which adds or updates information on the courses based on the information available from the service providers. The introduction of reinforcement learning allows the recommender system to dynamically respond to updates in the course catalog without retraining, which has not been done in other research works.

In [121] MARL is applied to recommendations in a social media setting, for potential mentioned (@'d) users in a tweet. It accomplishes this using an actor-critic method with two agents as actors. One agent evaluates historical tweets from the user, while the other agent evaluates tweets made by particular candidate mentioned user. These agents cooperate to form their recommendation, and the critic gives them feedback based on how good their recommendation was. Zhang et al. [351] provides another example of agents cooperating to provide a recommendation. In this work, the problem of recommending the best charger for an electric car based on proximity, waiting time, and other factors is considered. Like [121], an actor-critic framework is employed. Each charging station is an agent and constitutes an actor. Based on the information they know about themselves and other agents, they make recommendations to the centralized attentive critic. The critic then generates a recommendation based on the action with the highest "bid".

Other settings that have used Multi-Agent Reinforcement Learning for Recommender Systems are websites, as in [220]. There, the authors apply Multi-Agent recommender systems to the task of web adaptation (changing a website based on the user's preferences) with two agents. One agent uses single-condition association rules and a second agent uses collaborative filtering algorithm. Another way to use Multi-Agent Reinforcement Learning is to recommend Internet of Things devices like in [105]. Here, the authors present an algorithm for improving the performance in an Internet of Things (IoT) environment. Each "thing" is represented by a cyber agent. Through collaborative filtering, the system recommends the user additional items based on items

they already like.

The MARL TLeague framework [291] demonstrates that modeling a Multi-Agent game with reinforcement learning method for each agent is not good, as it leads to policy-forgetting. When agents are independent, they tend to rotate over the Nash Equilibrium. With fictitious self-play, the agents play not only against their present opponents, but historical opponents as well, which has the benefit of avoiding policy-forgetting as well as being able to reach Nash Equilibrium.

Another case against independent Reinforcement Learning comes from [169], which provides evidence that with independent reinforcement learning, there is much overfitting. One metric for judging how well a policy generalizes is Joint-Policy Correlation. The higher the correlation the more overfitted the policy is likely to be. This paper generalizes the same notion of Fictitious Self-Play present in [291].

More examples of the advantages of self-play can be found in [26, 77] In a Multi-Agent setting, the other agents can learn. This makes the environment non-stationary, and emphasizes the importance of learning for each agent. Two minimal properties of a good Multi-Agent reinforcement learning algorithm are the ability to learn the optimal policy against an opponent that is stationary or may become stationary, and convergence to the Nash Equilibrium if all agents have the same policy. to achieve these goals, the authors propose AWESOME: an algorithm that has agents adapt their strategies when their opponents appear stationary, and retreat to an equilibrium when opponents are adapting.

The competition amongst agents fostered by techniques such as self-play can result in interesting emergent behavior, as in [26]. In this work the authors observe that when multiple agents trained using self-play are put into simple environments, it is possible for complex behaviors to

be learned. In particular, self-play sees great results in competitive environments. Rather than constantly playing against the most recent version of an opponent, select a historical version of the opponent at random so one agent does not run away with the lead and dominate everyone else. Different policies can also be selected at random to avoid overfitting.

Chen et al. [66] survey serves to give me a review of the state-of-the-art in recommender systems that employ deep learning, including a problem statement for the task of making recommendations to a user, as well as overviews of sophisticated strategies for the different tasks encompassed by this paradigm using Multi-Agent Reinforcement Learning which is capable of handling multiple tasks. Feng et al. [99] addresses two weaknesses that happen when scenarios are optimized individually: there is no collaboration during the optimization and there is no way to model correlation between the optimizations. To this end, the authors present Multi-Agent Recurrent Deterministic Policy Gradient (MA-RDPG). The work makes the point that even in non-cooperative scenarios, collaboration between agents is necessary to maximize total return. Carroll et al. [54] studies how to predict the affect of different policies for Reinforcement-Learning based Recommender Systems on their users' preferences. To observe how a new policy influence its users, it is necessary to have access to a set of historical data on how the users interact with the old policy.

Difference from Prior Work

While there are several works that touch on the idea of making recommender systems explainable [52, 203, 353], mine is the first to consider the dynamics of treating AI explanations as the item to be recommended. I furthermore distinguish my work from others by modeling different users as unique agents in a Multi-Agent system that implicitly collaborate and negotiate

with each other via a shared reward.

8.2.5 XAI Methods and their Properties

I apply Multi-Agent Reinforcement Learning to the novel task of recommending XAI outputs from feature attribution methods such as DeepLIFT SHAP, Integrated Gradients, LIME, and Feature Ablation. These four methods were chosen for being among the most popular XAI techniques, and for having unique properties that make them suitable as a basis for studying how they are recommended in a variety of use cases. DeepLIFT SHAP uses a distribution of background samples, whereas Integrated Gradients uses a single baseline input. LIME is a stochastic method that trades consistency for reduced complexity, but Feature Ablation remains consistent while trading off for increased latency.

DeepLIFT SHAP

Shapley Additive Explanations (SHAP) [197] is an explainable AI attribution method that attempts to unify several different ways of explaining the feature importance for elements of an input to a model. SHAP takes inspiration from Shapley values of Game Theory [272] whereby the contribution of an individual feature can be estimated by seeing how the behavior of the model changes when that feature is replaced by a baseline.

It can be shown that SHAP explanations are locally accurate, meaning the internal interpretable representation of the model has the same output for data examples given to the original model. SHAP explanations are also consistent, meaning features that cause a large difference in prediction are important, and exhibit missingness, meaning an absent feature has zero importance.

Calculating the true Shapley values for each feature is intractable for many model inputs, so I explain AI outputs with DeepLIFT SHAP, a modified version of SHAP based on DeepLIFT [276]. DeepLIFT SHAP provides an efficient way to approximate SHAP explanations.

Integrated Gradients

Integrated Gradients is another feature attribution, especially useful for computer vision, but applicable to a wide range of scenarios. It exhibits two desirable properties:

Sensitivity: An attribution method has sensitivity if for any two examples that differ in one feature but have different predictions, that feature has a non-zero attribution value.

Implementation Invariance For any two networks that are functionally equivalent (i.e. given any valid input they produce the same output) but differ in implementation, their attributions should be equal. Put more formally, I have that an attribution method h is *implementation invariant* if

$$\forall x, f(x) = g(x) \implies h(f, x) = h(g, x) \quad (8.1)$$

for any two networks f, g .

LIME

LIME, or Locally Interpretable Model Explanations [250] is a feature attribution method that works by creating an interpretable explanation model (e.g. a linear model) that suffices to explain the behavior of a more complex model localized to the area around a particular example.

The example is projected to a simplified input space to aid the interpretability of the surro-

gate model. Samples are randomly taken from around the area of the simplified example, and the surrogate model's behavior is used to create feature attributions for the original model.

Feature Ablation

Feature ablation is a simple and straightforward attribution method. As described in [211], the approach is to compare a model's output with all the features of its data to the model's output with a baseline in place of a particular feature and use the difference between the outputs as a measure of that feature's importance. Described more formally I have

$$\phi_i(f, X) = f(X) - f(X_{D \setminus \{i\}}) \quad (8.2)$$

where i is the feature whose importance is being measured, X is an example from a dataset, f is the model, and D is the set of features, of which i is a member.

Feature ablation is a naive method of conducting feature attributions and as such does not exhibit any of the above mentioned XAI properties. It is instead a very approximate way to estimate the importance of features for a model's outcome.

8.2.6 Multi-Agent XAI Recommender System

RecSys with 4 XAI methods

The items that my Multi-Agent Recommender System serves to the user agents are AI explanations of the predictions from a Neural Network trained to perform a regression task using a dataset. For each example in the dataset, explanations are creating using DeepLIFT SHAP, LIME,

Integrated Gradients, and Feature Ablation. These explanations, which I define as “documents”, are compiled into a single collection, as seen in Figure 8.3. Each document has three user-observable features that the user agent can use to make their decisions on which XAI output to choose: latency, infidelity, and sensitivity. Attribution scores are not included as an observable feature of the XAI output so the user is not biased when making a decision.

XAI metrics

When considering which XAI outputs to put in the slate recommended to the user, my recommender agents observe three features for each document: latency, infidelity, and sensitivity. These metrics cover the trade-off between robustness and time cost commonly seen in real-life deployments of AI Applications [323] and serve a starting point that can be expanded in future work.

Latency: Simply put, latency is the amount of time it takes to create a particular XAI output. During the generation of the XAI outputs, I note the time immediately before and after each output and take the difference to be the latency:

$$\text{LAT}(\Phi, f, x) = \text{end} - \text{start} \quad (8.3)$$

where Φ is the attribution method, f is the model being explained, x is the input, end is the time when the attribution method stopped running, and start is the time when the attribution method started. When I generate XAI outputs for LIME, I choose the hyperparameter for the number of samples created by LIME to be 5 from the default value of 50 to have a more equitable comparison with the other methods.

Infidelity: The notion of infidelity as presented in [339] seeks to capture a score for what happens when, given an explanation method, a model, an input, and a set of perturbations, what happens to the explanations relative to the differences between the model with the original inputs and the perturbed inputs. For this the proposed score is mean squared difference between an explanation multiplied by a set of perturbed inputs and the difference between the model output with the original inputs and the model output given the perturbed outputs. To put it more formally, given attribution method Φ , model f , input \mathbf{x} , and perturbation matrix \mathbf{I} , I have

$$\text{INFD}(\Phi, f, \mathbf{x}) = \mathbb{E}_{\mathbf{I} \sim \mu_{\mathbf{I}}} [(\mathbf{I}^T \Phi(f, \mathbf{x}) - (f(\mathbf{x}) - f(\mathbf{x} - \mathbf{I})))^2] \quad (8.4)$$

.

\mathbf{I} can take any number of forms as a set of perturbations. For my purposes, I take \mathbf{I} to be a noise vector with elements randomly sampled from a zero-mean normal distribution.

SensitivityMax: The idea of sensitivity, similarly to infidelity, aims to serve as a quantitative measure for how much an explanation varies with small differences in the input vector. One way to measure this according to [339] is *SensitivityMax*. *SensitivityMax* aims to find an input within an infinity-norm ball of radius r about an original input in the feature space that maximizes the difference between the explanation at those inputs:

$$\text{SENS}_{\text{MAX}}(\Phi, f, \mathbf{x}, r) = \max_{\|\mathbf{y} - \mathbf{x}\| \leq r} \|\Phi(f, \mathbf{y}) - \Phi(f, \mathbf{x})\| \quad (8.5)$$

.

Agentic RecSys

Each recommender agent for my Recommender System uses the SlateQ reinforcement learning algorithm [144]. SlateQ is an extension of the concept of Q-learning [324], but while normal Q-learning maps values to each possible action, SlateQ maps Q-values to combinations of items, known as *slates*. The agent has a Deep Q Network that assigns a Q-value to each possible slate and then recommends either the slate with the highest Q-value or a random slate depending on whether the Deep Q Network is set to exploit its current knowledge of the users preferences or explore the user's preferences more.

User State

The user state consists of the user's preferences for each of the three XAI metrics (latency, infidelity, and sensitivity) and the user's time budget, which determines when the user will stop using the recommender system. If the time budget is at or below zero when a step ends, the user terminates. If all users have terminated, the episode ends, another one is started, and the user reappear with their default settings. The user states per episode can be dynamic depending on the time budget and whether the recommended items have any utility to the users' preferred goals. Implicit negotiation between users with different goals is reflected when each user indicates their satisfaction with the XAI output and their responses are combined to form a unified reward signal. An illustration of the user feedback process can be found in Figure 8.4

To make decisions, each user employs a Multinomial Proportional Choice Model, represented by the "Choice Models" in Figure 8.3. The Choice Model calculates scores for each document in the slate according to $\text{score} = \mathbf{u} \cdot \mathbf{d}$. After the scores for the documents in the slate are generated, a document (represented by the "Selection" in Figure 8.3) is chosen from the slate

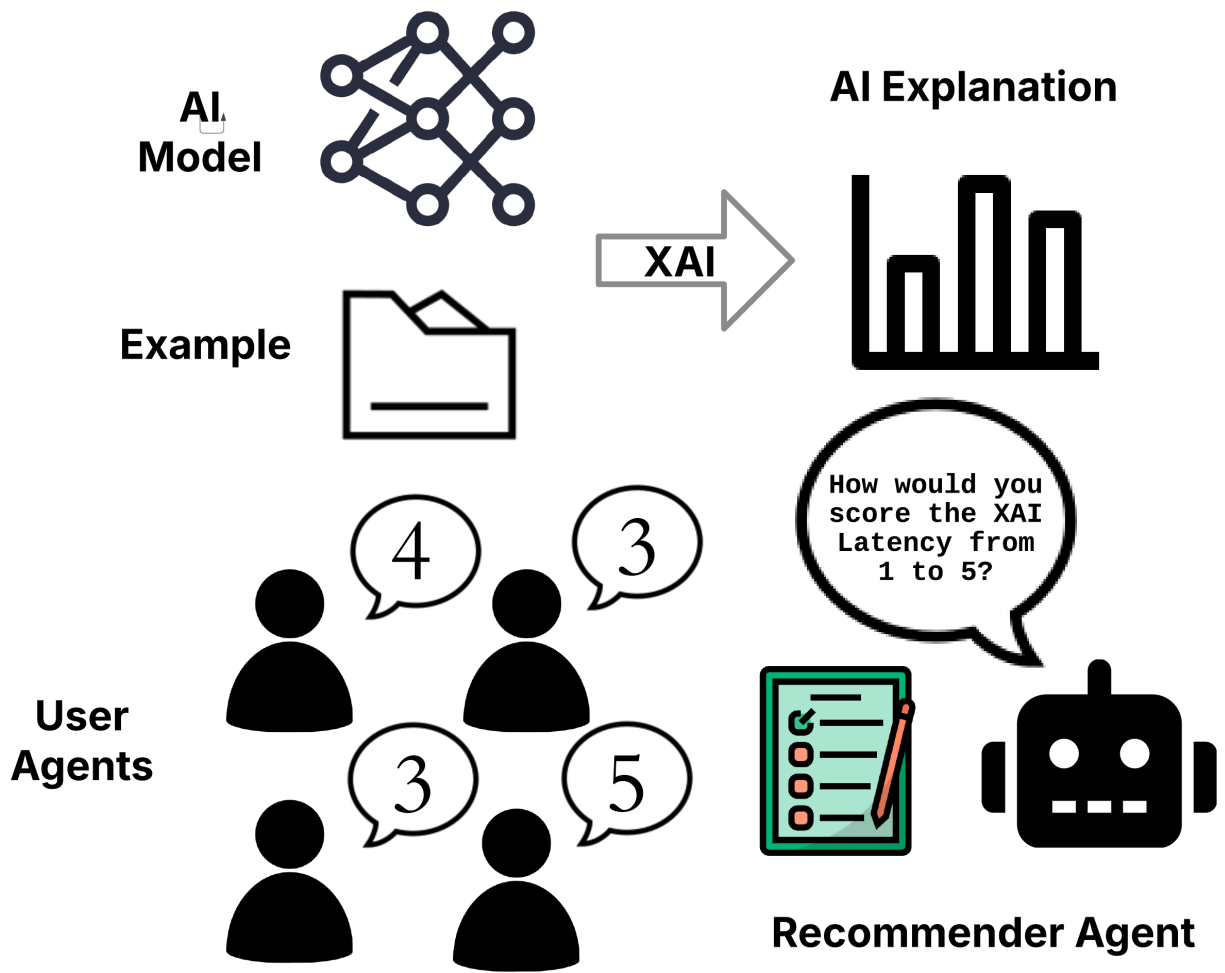


Figure 8.4: Implicit Negotiation between User Agents for XAI Outputs of AI Models

with a probability

$$p(x) = \frac{x}{\sum_{x \in X} x} \quad (8.6)$$

where x is the score of the document and X is the set of all document scores. Once a document is chosen, the alignment and quality of the response are calculated using equations 8.7 and 8.8 respectively. As seen in Figure 8.3, the response quality is then used as input to the reward function that will provide feedback to the recommender agent, and the user transition model is called to update the user preferences and time budget.

User Transition Model

The user's preferences and time budget (the number of steps the user will continue to run) get updated by the user transition model after each step, represented by the "Transition Model" text in Figure 8.3. The user's evolving preferences depend not only on their current preferences but their past preferences in a self-play mechanism. The update to the user preferences is randomly initialized. At this point, take the observable features of the XAI output, a step size calculated based on the user interests, and multiply the difference times the XAI output observation to get an update vector. Then take the dot product of the XAI output observation with the user preferences. The result is compared with a random number between 0 and 1 . If the dot product is greater than the random number then the update is added to the current user preferences. Otherwise the update is subtracted. The resulting vector is rounded to the nearest integer and then clipped to the $[1, 5]$ so it is guaranteed to be within the original Likert scale space.

After each step, the time budget decrements by one and then updates according to the quality of the document chosen by the user. After the quality is scaled by a time budget step size

hyperparameter (here, 0.1) it is added to the time budget. In this way, the higher quality recommendations the user receives, the longer they are incentivized to interact with the recommender system.

Reward Formulation

The reward for the multi-user recommender agent, represented by the purple block in the far right of Figure 8.3, is a function of how well the user’s preferences on the XAI metrics align with the values for those metrics on the XAI outputs they have chosen. For example, if the user cares a lot about latency and the XAI output has a low latency then the recommender agent will receive a higher reward.

A comparative analysis of the recommender system on two different reward functions is given. Each reward function takes as input the observed user preferences, the observed document features, and the responses from the users to the documents that they were recommended by the recommender agent. The user only chooses to interact with one XAI output at each step so only the information from the response to the chosen document is considered when formulating a reward. Based on the user preferences and the features of the document the user chose to interact with, the user response to the document has an *alignment* with the user preferences. This alignment is modelled as the cosine similarity between the user observation vector and the document observation vector:

$$c(\mathbf{u}, \mathbf{d}) = \frac{\mathbf{u} \cdot \mathbf{d}}{\|\mathbf{u}\| \cdot \|\mathbf{d}\|} \quad (8.7)$$

where \mathbf{u} is the observation vector of the user and \mathbf{d} is the observation vector of the XAI output. The cosine similarity is adjusted slightly to account for cases where the XAI output not

only meets but outdoes the needs of the user. This is accomplished by rounding the alignment of the response up to one when most features of the document are greater than the features of the user:

$$q(\mathbf{u}, \mathbf{d}) = \begin{cases} 1, & \text{if } \sum_{i=1}^n \mathbb{1}_{d_i - u_i > 0} > \frac{n}{2} \\ c(\mathbf{u}, \mathbf{d}), & \text{otherwise} \end{cases} \quad (8.8)$$

In this way the recommendation response *quality* is not penalized for not aligning with the user's preferences if the document is generally better than the user's preferences. After calculating the quality of the response from the user to the document, the responses of each user are passed to a reward function. The two reward functions used are Value-Decomposition-based [293] rewards and QMIX-based [248] rewards.

QMIX is an off-policy action-value function factorization algorithm that learns a joint action-value function $Q_{tot}(\boldsymbol{\tau}, \mathbf{u})$ to develop consistent centralized and decentralized policies. $\boldsymbol{\tau}$ denotes the joint action-observation history and \mathbf{u} as the joint action. Other action-value function factorization algorithms, such as Value Decomposition Networks (VDNs), compute the joint action-value function for a centralized policy by taking the sum of each individual action-value function $Q_i(\tau_i, u_i, \theta_i)$. QMIX argues that this full factorization is not required to maintain consistency between the centralized policy and each decentralized policy, only as long as the arg max of the joint action-value function is the same as the arg max of each individual action-value function

[248].

$$\arg \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \arg \max_{u_1} Q(\tau_1, u_1) \\ \vdots \\ \arg \max_{u_n} Q(\tau_n, u_n) \end{pmatrix} \quad (8.9)$$

QMIX and VDNs are used for the basis of the reward functions by supplying the response quality and states to the networks and receiving a value, which is used as the reward. For this the implementation of QMIX and VDNs from [234] is used.

Multi-Agent XAI RecSys

During the episodes of the recommender system, there are multiple users being recommended XAI outputs. Each of the users have their own unique preferences and goals that shape how they respond to the recommendations they receive. These users may interact with each other through a shared reward signal, seeing how their goals may align or differ so their features can be further refined to optimize their recommendations. An example of this may be a potential homeowner and a bank interacting with each other to see how different variables affect a model used to predict housing prices in a neighborhood. These users can have very different interests; the homeowner may care more about how sensitive the explanations are to small changes in the input while the bank may want to run many attributions quickly, but if they share information by interacting with each other the recommendation policy can be tweaked to benefit both of them.

Evaluation Metrics

There are several metrics (as seen in Figure 8.5) to evaluate the performance of the Multi-Agent Recommender System in the absence of real-world user data.

Average Alignment represents the average (avg) value of the alignment (as calculated in equation 8.8) between the users' preferences and the documents they chose at each time step. This, along with the other metrics listed here, is calculated over each iteration.

Recommendation Diversity is useful to measure how the 4 XAI methods were recommended items per user. It helps to get an insight whether increasing the slate size or using QMIX or VDN algorithms increases the diversity of recommendations.

Average Episode Length is the average length of each episode within an iteration which is a measure of how well exploration happened with the QMIX and the VDN algorithms for different slate sizes. Each user has a *time budget* that determines how long they participate in an episode. This time budget decrements by one at each step, but also increases by $0.1 \times quality$ at each step, where *quality* is the quality of the user's chosen document. Thus, a longer episode means the users were recommended more higher-quality documents.

Average Quality measures the average document quality as calculated in Equation 8.8 for the documents chosen by the users.

Average Click-Through-Rate is the ratio of the number of times a document was recommended to the user and the number of times it was chosen.

Average Episode Reward. This metric shows the average of the sum of rewards for each step in each episode.

Average Standard Deviation of Episode Rewards is the average standard deviation of the rewards for the episodes in an iteration.

Time Per Step is simply the average amount of time it takes for one step to complete within an iteration.

Results

Experiment Setup All experiments were run on a NVIDIA GeForce RTX 2080 Ti GPU with 32GB of memory. The operating system used was Red Hat Enterprise Linux 8. Libraries used include recsim 0.2.4, pandas 2.2.2, and tensorflow 2.15.

The most salient findings from the experiments are that recommending a slate of items instead of a single item results in improving recommendation diversity for different users, along with an increase in alignment between the features of the chosen XAI output and the user preferences when combined with a QMIX or VDN-based reward. For these experiments, the training loop ran for 20 iterations with the number of training steps for each iteration capped to 10000. These experiments ran using the standard SlateQ-based recommender agents and compared it with a non-agentic random baseline with randomly selected items in slates. Slate sizes of two and one are contrasted for comparative analysis on introducing slates for recommendation diversity in Q-learning based recommenders. For 4 XAI methods, other slate sizes like three would be redundant, having the same combinatorial space as the slate size of one ($\binom{4}{1} = \binom{4}{3} = 4$). A slate size of four is akin to no recommendation being made at all, as there are four XAI methods to consider. Thus, all possible slate sizes are explored.

There are two different reward functions: a reward based on the output of a QMIX network and a reward based on Value Decomposition Networks (VDNs). The loss function for the SlateQ recommender agents was standard Huber loss. Figure 8.7 shows that when using a QMIX-based reward and a slate size of 1, each user tends to gravitate towards a "favorite" XAI method that dominates the rest of the distribution. However, compared with a slate size of 2, the other XAI methods being chosen at a more variable rate.

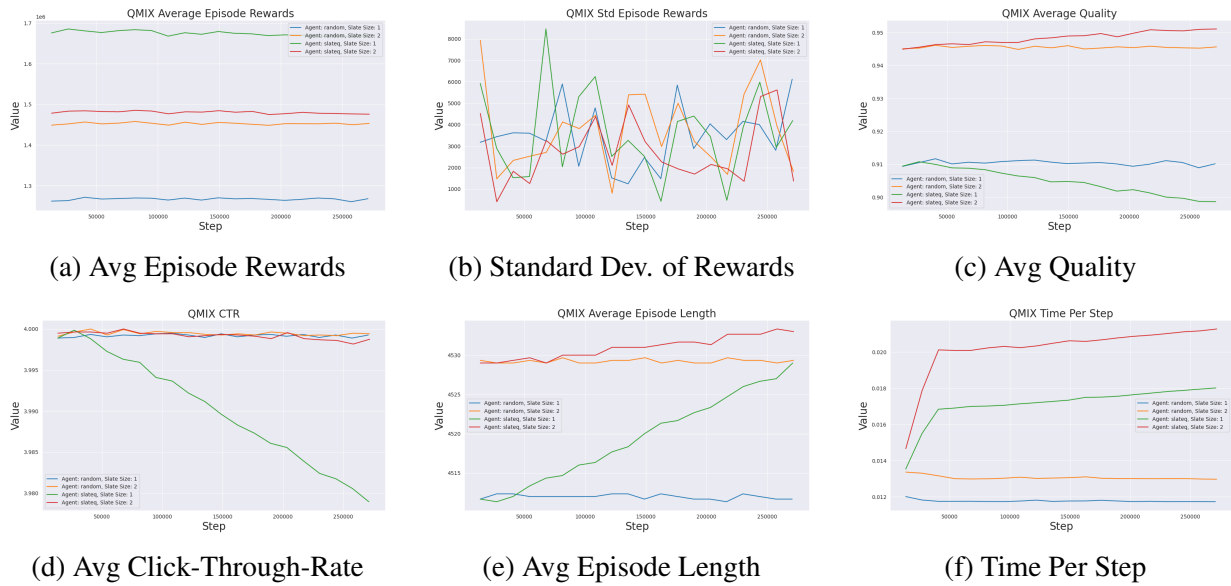


Figure 8.5: Evaluation Metrics for QMIX Agents and Random Slates

Figure 8.6 shows that with more training steps, the average alignment of the chosen XAI document with the user preferences increases for both QMIX reward and VDN reward when the slate size is 2, although the alignment is slightly higher when the QMIX reward is being used, with the maximum average alignment with the QMIX reward approaching 0.900 compared to the maximum average alignment with the VDN reward closer to 0.896.



Figure 8.6: Avg Alignment for QMIX and VDN in comparison to Random strategies

Figure 8.8 in the Appendix compares the results for experiments run with a VDN reward for SlateQ agents and Random slates with slate size 1 and 2. Comparing the average episode

rewards for VDN in contrast to the QMIX reward, the VDN reward is lower than the reward for the random baseline, showing that QMIX is a better overall reward for this use case.

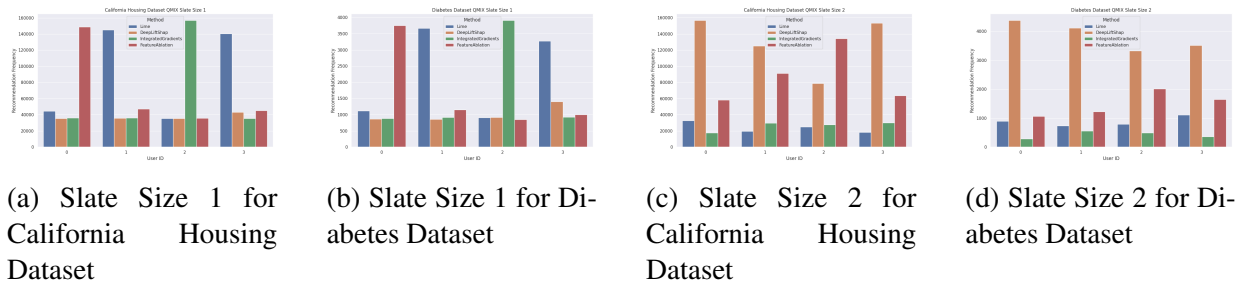


Figure 8.7: Recommendation Diversity with QMIX Agents

In Figure 8.5 and Figure 8.8 over time, the reward generally increases for both QMIX and VDN-based rewards, showing that these rewards lead to effective learning of user preferences over time, resulting in better recommendations.

Another interesting result from the experiments is that the time it takes for each step of the recommender system to execute is seemingly correlated with the size of the slate recommended to the user, implying that when a large number of simulated user interactions is needed quickly, a smaller slate size is better.

8.2.7 VDN based Recommendation Results

According to Figure 8.8 as opposed to the QMIX-based reward function whose rewards are shown in Figure 8.5, the VDN reward starts off much lower than the random baseline, though it does trend upward, indicating that learning is occurring. Also, while the Episode Length is completely stagnant for the random baseline, the episode length increases almost monotonically for the SlateQ agent with the VDN reward, indicating more exploration is happening, leading to the discovery of better recommendations.

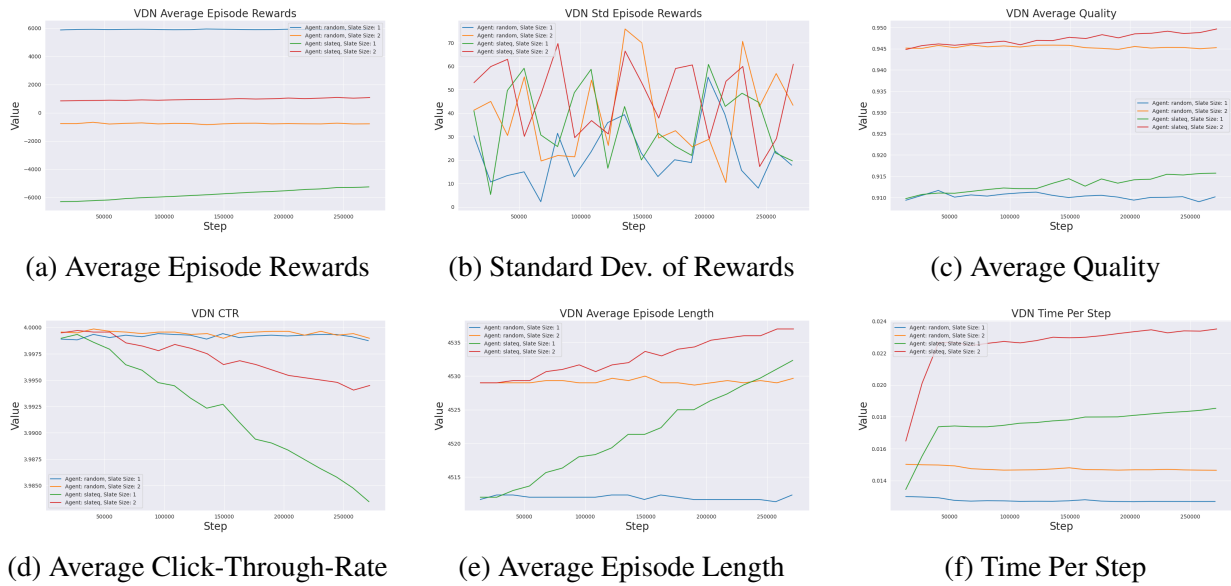


Figure 8.8: Evaluation Metrics for VDN Agents and Random Baseline

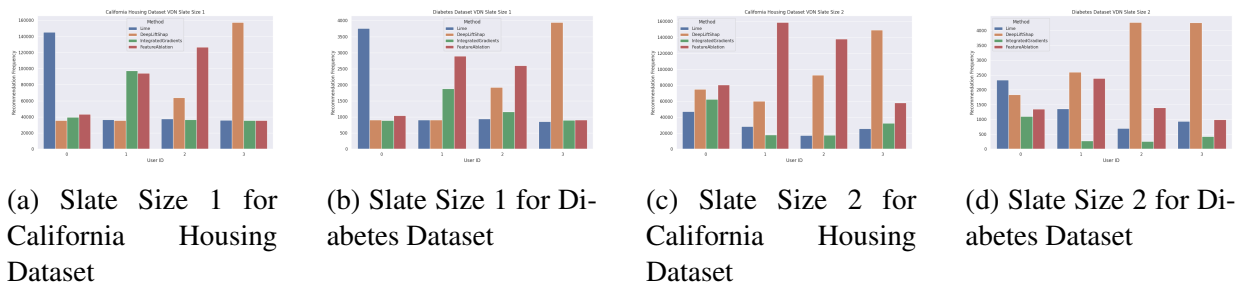


Figure 8.9: Recommendation Diversity with VDN Agents

From Figure 8.9 it can be surmised that, like QMIX, the VDN-based reward also results in diversity for user choices, especially when the slate size is two. VDN favors each attribution method for at least one user.

8.2.8 Conclusions and Future Work

My research on Multi-Agent Recommender System (RecSys) for XAI methods is aimed to support XAI researchers and users of AI models to understand which XAI method they should use that satisfied their goals of a fast and/or robust method for 4 XAI methods: DeepLIFT

SHAP, LIME, Integrated Gradients and Feature Ablation. The proposed Multi-Agent XAI RecSys framework is a Collaborative Interactive RecSys (CIR) which models preferences of four user agents' sharing mutual interests along XAI features like latency, sensitivity and infidelity using a Likert scale representation and shapes the reward function according to the users' preference. A Slate-based Multi-Agent CIR trained on the QMIX algorithm is used to observe higher alignment of the recommended features to user goals and more diverse recommendations. The user preferences are updated with a self-play approach considering the current preference for a ML use case with preference updates based on observing previous use cases. Comparison with the VDN algorithm shows that QMIX algorithm is more representative of the goal preferences of multiple users sharing mutual interests as it improves higher alignment, more diversity, and more user engagement with the recommended users. Both QMIX and VDN based Multi-Agent RecSys has higher quality of diverse recommendations than random strategies, indicating the importance of Multi-Agent modeling of the XAI RecSys.

Future work involves collecting human preferences instead of simulated preferences to extend the reward shaping of the Multi-Agent Reinforcement Learning framework with human feedback and reflecting the housing and banking use cases with humans in the real world. More research can be done with Independent Q Learning which faces convergence challenges in existing Multi-Agent RL (MARL) research to investigate if similar challenges exist in this XAI RecSys framework. Further comparative analysis can be done with policy-based MARL algorithms like MAPPO, MADDPG among others or addressing the stability challenges of QMIX with weighted QMIX. Adding more XAI features like reliability can help to improve trustworthiness with Explainable AI Agents that explore whether an XAI method can identify expected influential features in attributing an ML algorithm's outputs for a particular task. If an XAI

method cannot identify these influential features for different examples in the same ML task, that knowledge can give an indication about whether the XAI method is reliable and trustworthy.

8.3 Explainable Visualizations

Past research on visualizing word embeddings across semantic dimensions at AAAI-22 [23] serves as a motivation to improve the visualization of XAI methods satisfying the objectives of different users with Multi-Agent reinforcement learning representations on which I've published relevant work [e.g., 25].

Word embeddings represent words as high dimensional feature vectors. Their use is now common in neural network NLP. In their seminal paper on word2vec, (Mikolov 2013) [212] showed that a feature space created by training a word prediction network on a large text corpus will include semantic information that supports analogy by vector arithmetic, e.g., "king" minus "man" plus "woman" equals "queen". To help novices appreciate this idea, people have sought effective graphical representations of word embeddings. Past research on word2vec demonstration involves the construction of a semantic space from collections of word pairs that contrast along one of two dimensions: gender (e.g., man vs. woman; king vs. queen) or age (e.g., man vs. boy; king vs. prince). Word vectors are then plotted in 3D, where the third "residual" dimension is roughly aligned with the contrast words but orthogonal to the first two dimensions. The interactive, browser-based demo presents a rotatable and zoomable display of this 3D semantic space and allows users to project new words into it. It can also display the 10 closest words to a selected word. In addition users can visualize and compare the actual 100-element values for selected words. This can lead K-12 students to valuable insights into the nature and limitations

of word embeddings.

8.3.1 Graphical Display of Embeddings

In some cases, semantic features in embedding vectors can be discovered by examining the vectors directly. Figure 8.10 shows the vectors for three male words and the corresponding female words. Element 76, marked with a black line, correlates well with gender. But things aren't always this simple; these words were chosen to give a clear example. Figure 8.11 shows vectors for three words associated with adulthood and three associated with youth. Component 14 correlates well with age, but again, these words were chosen because they offer a clear contrast. Semantic information is more commonly distributed across multiple vector components and is difficult to discern by looking at the vectors directly.

Displaying the vectors directly is still useful for some purposes, but does not help the user appreciate the geometry of the semantic space that makes analogy by vector arithmetic possible. If each word is a point in the space, direct display of the vector for that word, as in Figures 8.10 and 8.11, is like writing out its coordinates. What is done instead is to *graph* the point.

8.3.2 Visualizations to Explore Word Embeddings

One of the core design principles for the demo is that it should display something interesting from the moment it's loaded, without requiring the user to take any action. So the already defined gender/age semantic space is used along with the addition of the vectors for a select group of words to the 3D scatter plot on the left. To the right, the 100-element vectors are displayed for six selected words: man, woman, boy, girl, king, and queen. Figure 8.12 shows the initial display

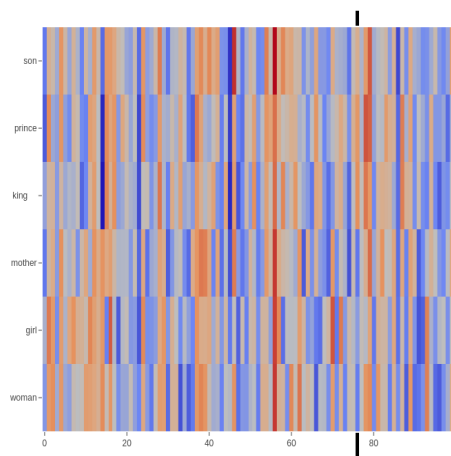


Figure 8.10: Embedding vectors for three male and three female words. Component 76, marked by a vertical black line, is positive for male words and negative for female words.

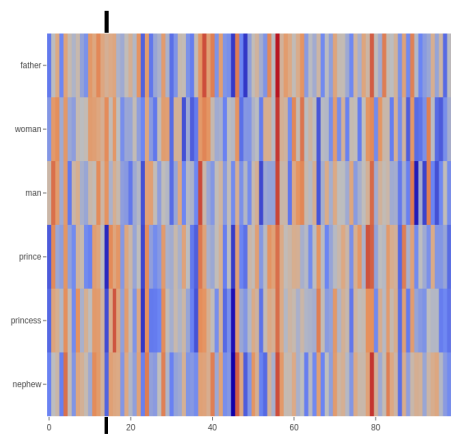


Figure 8.11: Embedding vectors for three words associated with adulthood and three words associated with youth. Component 14, marked by a vertical black line, is positive for the adult words and negative for the youth words.

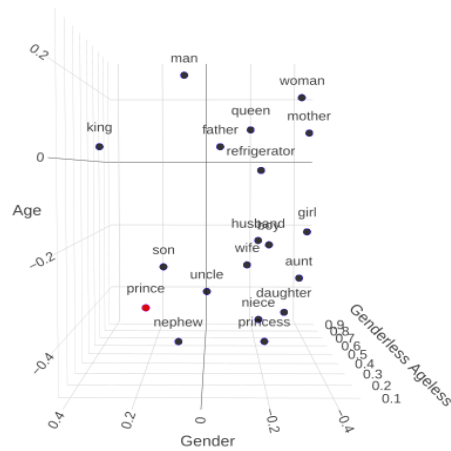


Figure 8.12: Words plotted in 3D semantic space. Male words appear in the positive (left) half of the x-axis; female words in the negative (right) half. Adult words are in the positive (top) half of the y-axis; youth words in the negative (bottom) half.

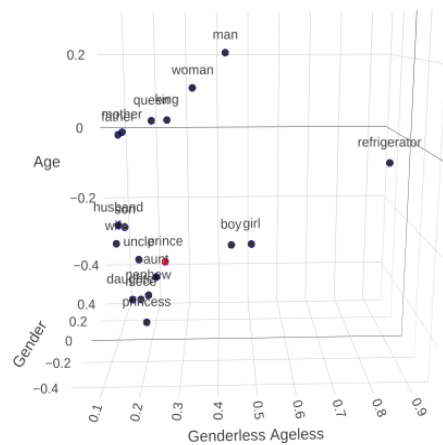


Figure 8.13: A rotated view of the scatter plot showing distance along the third, residual dimension, which is labeled Genderless/Ageless. The contrast pair words, which define the residual dimension, have residuals close to zero. “Refrigerator”, which is semantically unrelated to the contrast words, is located a substantial distance from them (high residual).

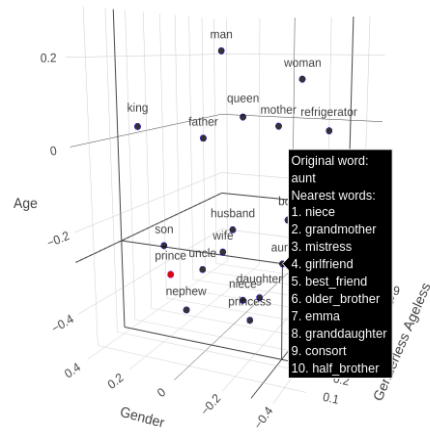


Figure 8.14: Hovering over a point in the scatter plot produces a display of the word and its 10 nearest neighbors.

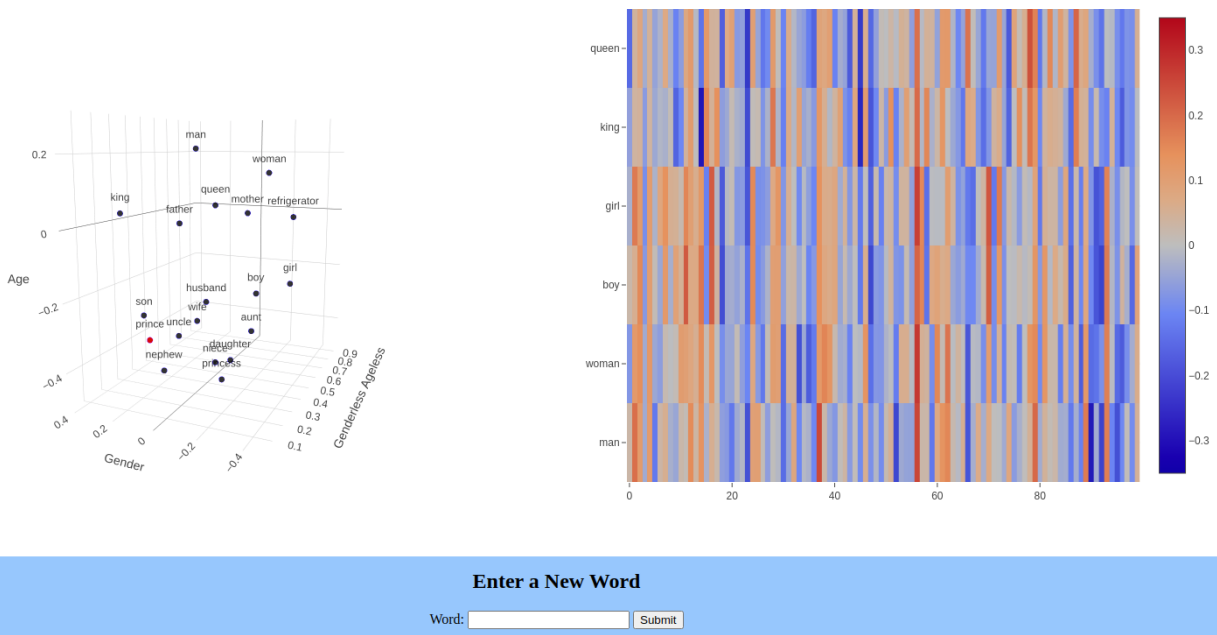


Figure 8.15: Layout of the developed interactive embedding visualization demo. On the left side is a zoomable and rotatable 3D scatter plot. On the right is a display of selected embedding vectors. The text box at the bottom allows entry of new words.

when the demo is loaded.

Continuing the theme of providing a low threshold for informative interactions, simply moving the mouse pointer over a point in the scatter plot produces a pop-up window displaying that word and its 10 closest neighbors (Figure 8.14), measured by 100-dimensional Euclidean distance. Exploring the neighbors illustrates how dependent the vectors are on the choice of training corpus. For the vectors that have been used, which were trained on Wikipedia articles, the closest neighbors of “apple” after “macintosh” are computer terms such as “amiga” and “microsoft”, not fruits.

Displaying the word vectors as a heat map in the right half of the window facilitates visual comparisons, but transparent about the values is also a major user requirement. Moving the mouse pointer over an element of the word vector display produces a pop-up showing the word, the position of the element in the vector (0 to 99), and the value of that element, which typically ranges from -0.35 to 0.35.

Clicking and dragging on whitespace in the 3D plot rotates the view, while the scroll wheel can be used to zoom in or out. This gives the user a better sense of words as points in a “space”, with relative motion of points providing a helpful cue for depth.

Clicking on a word causes it to become “active”, and its dot turns red. Only one word can be active at a time. When a word is active, clicking on a row of the vector display in the right half of the window causes that word’s vector to be copied into that row. In this way, the six words initially shown in the vector display can be replaced with words of the user’s choice. This makes it easy to visually compare vectors of related or contrasting words, as shown in Figures 8.10 and 8.11.

New words can be added to the scatter plot by typing them into a text box at the bottom

of the window. When a word is entered, it is made the active word; the red dot helps the user to locate it in the display. A previously-added word can also be entered in the text box to make it active again if the user needs help finding it.

8.3.3 Conclusion & Takeaways to Visualize XAI Outputs

The visualization research to explore and learn word embeddings is relevant to the domain of XAI, specially focusing on explaining text based or multimodal AI models. Current research in XAI is mostly reliant on visualizations to understand the importance of features post-hoc after training, be it for the SHAP, LIME, Integrated Gradients or other XAI methods. While recommending XAI outputs is an important research goal to build trustworthy AI, it is also a big focus to improve the XAI visualizations with better representations of underlying features along with providing explainable visualizations that can be projected across semantic dimensions based on the needs of AI users. There is flexibility for the users to define their own semantic dimensions.

8.4 Target Representations in Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) algorithms have been successfully applied to real world situations like illegal smuggling, poaching, deforestation, climate change, airport security, etc. These scenarios can be framed as Stackelberg security games (SSGs) where defenders and attackers compete to control target resources. The algorithm's competency is assessed by which agent is controlling the targets. I have investigated the modeling of SSGs in MARL with

a focus on possible improvements of target representations in RL algorithms [25]. The objective is to have interpretable target representations which can be easily explained to end-users of AI in order to build trust in AI models.

8.4.1 Introduction

Multi-Agent Reinforcement Learning (MARL) is useful to solve real world problems whose models are inspired by game theory (GT). Stackelberg security games (SSGs) in GT formulate real world agents competing to control targets in domains like protecting infrastructure such as airports, trains, traffic, etc. [278], preventing drug smuggling, combating illegal fishing and animal poaching, mitigating illegal logging, and other climate change problems [68]. Here is an investigation into how MARL algorithms model SSGs with a focus on understanding target representations to get efficient and explainable solutions. Successful automated assessment of RL algorithms for SSGs require well-defined and interpretable target (secure resources) representations. Targets (secure resources) over which attacker and defender agents are competing have to be represented and understood properly to assess the success of policies in autonomous systems. To make the automated assessment concrete on the basis of targets, the definition and representation of targets have to be clearly understood.

8.4.2 Stackelberg Security Games

Stackelberg security games (SSGs) are two-player games with one defender and one attacker. The defender aims to protect a set of targets T by allocating limited resources to protect up to b targets, where $b \subset T$. The payoff functions for the defender and attacker are $U^d(t)$

and $U^a(t)$ respectively for $t \in T$. \mathbf{p}_t is the probability of target t to be protected by the defender and $\mathbf{q}_t(\mathbf{p}, \xi)$ is the probability of target t being attacked by the attacker with feature ξ . [316]. The two-player SSG has been extended to a coalitional security game [124] or single leader multiple follower SSG by using a cardinal function of agent utilities [68]. For a single leader, multiple follower Stackelberg security game, the utility function for the leader (defender) is $U_l : \Omega_l \times \Omega_f \rightarrow \mathbb{R}$. The utility function for a follower (attacker) is $U_i : \Omega_l \times \Omega_i \rightarrow \mathbb{R}$, where Ω_l is the strategy set for the leader, Ω_i is the strategy set for the follower, and Ω_f is the combined strategy sets for all followers [68].

8.4.3 Targets for Competency Self-assessment

Traditionally, the performance of the autonomous agents in MARL can be evaluated using reward signal such as Bellman equation [306]. [163] proposes a novel technique for testing the robustness of MARL agents where “unit tests” are designed to gauge the agents’ abilities to solve specific tasks. A Strong Stackelberg Equilibrium (SSE), where the attacker breaks ties in the best response set in favor of the defender, is the standard solution concept for SSGs [312] due to no guarantees about the existence of a Weak Stackelberg Equilibrium (WSE) [78]. Even the SSE may not be applicable for security games, as the defender may not induce the attacker to select a preferred action with an infinitesimal change in their policy due to resource assignment constraints. [125] proposes the inducible Stackelberg equilibrium (ISE), defining an inducible target that serves as the lower bound of the defender’s utility guarantees. SSGs involve targets that the attackers attempt to gain control over and the defenders protect from the attackers [158, 279, 316]. The control of targets determines the winner, consequently determining the success of

policies in autonomous systems that help defenders to win. This motivates future work to develop a solution concept for SSGs with the notion of targets that can be applied to RL since control of targets by the attacker implies that the game is lost while the opposite implies a defender’s win.

8.4.4 Modeling SSGs in Reinforcement Learning

Markov Decision Process

Markov Decision Processes (MDPs) are described by a tuple of (S, A, P, R) where S is the set of states, A is the set of actions, $P : S \times A \rightarrow \Pi(S)$ is the state-transition function where $P(s'|s, a)$ is the probability of transitioning from state s to s' given action a , and $R : S \times A \rightarrow \mathbb{R}$ is the reward function where $R(s, a)$ is the reward of action a in state s [29].

For a sequential stochastic SSG with one leader (defender) and one follower (attacker), [214] proposes the Markov Perfect Stackelberg Equilibrium (MPSE), which computes a Markov equilibrium in linear time from a Stackelberg equilibrium in double exponential time using sequential decomposition, and applies it to a MDP. A model-free RL algorithm motivated by Expected Sarsa computes the optimal MPSE strategies for the defender and the attacker, where the defender maximizes its returns based on the knowledge that the attacker is strategizing to play the best response to the defender’s strategy.

Uncertainty in MDPs The transition probabilities in MDPs can be uncertain due to a lack of knowledge about true transition leading to uncertain rewards as formulated in Uncertain Markov Decision Process(UMDP) [62, 251]. [334] models SSGs as UMDPs using double oracle and deep deterministic policy gradient (DDPG) to find optimal defense strategy against poachers while maximizing regret.

[214] applies the idea of computing Markov equilibrium in linear time from Stackelberg equilibrium in double exponential time to a Markov Decision Process referred to as the Markov Perfect Stackelberg Equilibrium (MPSE). [161] applies MDP to a 2D grid game with 2 non-cooperative players. [278] looks into real-life problem of airport passenger screening.

Uncertain Markov Decision Processes (UMDPs) capture the uncertainty caused by the transition probabilities. It is defined as a set of tuple (S, s_0, A, T, R) where S is the set of state, s_0 is the initial state, A is the set of actions, $T = T_1, \dots, T_{|\xi|}$ is the set of uncertain transition functions and $R = R_1, \dots, R_{|\xi|}$ is the set of uncertain reward functions. A sample of model uncertainty is defined as $\xi_q = (T_q, R_q)$ where $T_q \in T$ and $R_q \in R$ [62, 251].

Existing papers model SSGs as UMDPs by introducing uncertainty in observations. [334] uses a double oracle and a deep deterministic policy gradient (DDPG) to find the optimal strategy to defend against poachers while maximizing regret.

Partially Observable Markov Decision Process

Partially Observable Markov Decision Processes (POMDPs) extend the MDP's definition to (S, A, P, R, Ω, O) where Ω is the set of observations and $O : S \times A \rightarrow \Pi(\Omega)$ is the observation function where $O(o|s', a)$ is the probability of the agent observing o after action a landing in state s' [152].

[7] models SSGs as POMDPs where agents cannot access the full information regarding the state. [273] finds the optimal defender policies with policy gradient in a partially observable cooperative RL framework. [311] uses Double Deep Q Learning (DDQN) to compute allocation and patrolling strategies with drones for SSGs with partial observability.

Dec-Partially Observable Markov Decision Process

Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) generalize POMDPs as $([N], S, A_{i \in [N]}, P, R, \Omega_{i \in [N]}, O_{i \in [N]})$. Considering an m -sized subset of N agents who can only partially observe the states, Dec-POMDPs update the set of actions A_i for agent i , the transition probabilities $P(s'|s, a_1, \dots, a_m)$ with $[1, m] \subset [N]$, the joint reward $R(s, a_1, \dots, a_m)$, the set of observations Ω_i for agent i , the observation function $O_i : S \times A_i \rightarrow \Pi(\Omega_i)$ with observation probabilities $O(o_1, \dots, o_m | a_1, \dots, a_m, s')$ that denotes the set of agents $i = 1 \dots m$ observing (o_1, \dots, o_m) after a tuple of actions (a_1, \dots, a_m) landing in s' . Dec-POMDPs formulate SSGs in Multi-Agent RL by modeling decentralized control over agents [31] by considering an m -sized subset of N agents who can only partially observe the states. It is parameterized by the set of agents $[N]$, the set of state S , the transition probabilities $P(s'|s, a_1, \dots, a_m)$ with $[1, m] \subset [N]$, the rewards $R(s, a_1, \dots, a_m)$, the set of observation Ω_i for agent i , the observation function $O : S \times A_i \rightarrow \Pi(\Omega_i)$ with observation probabilities $O(o_1, \dots, o_m | a_1, \dots, a_m, s')$ that denotes the of agents $i = 1 \dots m$ observing (o_1, \dots, o_m) after a tuple of actions (a_1, \dots, a_m) landing in s' . Dec-POMDPs are useful to formulate SSGs in Multi-Agent RL with the concept of decentralized control over agents [31].

Partially Observable Stochastic Games

Partially Observable Stochastic Games (POSGs) generalize from Dec-POMDPs to $([N], S, A_{i \in [N]}, P, R_{i \in [N]}, \Omega_{i \in [N]}, O_{i \in [N]})$. Its transition function is updated as $P : S \times \prod_{i \in [N]} A_i \times S \rightarrow [0, 1]$; It defines reward functions for each agent i as $R_i : S \times \prod_{i \in [N]} A_i \rightarrow \mathbb{R}$. and observation function as $O_i : A_i \times S \times \Omega_i \rightarrow [0, 1]$. POSGs formulate the Multi-Agent environment by

simultaneous stepping where all agents step, observe, and are rewarded together [299].

8.4.5 Target Representation to Assess RL for SSGs

Targets are important in SSGs, signifying the essential component of security: Attackers try to gain control from defenders while defenders protect targets from attackers. The ways targets are modeled in RL literature for SSGs are analyzed.

Targets as a Part of State

[273] defines targets to be a part of the state, while the objectives for the defenders and attackers are to protect and attack the targets respectively given a state. However, targets can be moved from one state to another [316], either as a result of agent actions or due to stochasticity. Existing research work applying repeated SSGs with imperfect information in RL [7] considers the targets to satisfy Markov and complex physical constraints where the games' strategies evolve around patrolling the targets which are fixed locations on a map. Likewise, the learning algorithms to solve security games consider the targets to be nodes or cells in a grid-world state space where vulnerable resources are located [161, 311, 321, 342]. [113] considers the targets to be a tuple of the location and time of the patrols protecting resources. These proposed algorithms do not define targets explicitly, which should be done for the solution of ISE to hold as defined before. [311] considers targets in SSGs as cells in the grid world environment with partial observability representing density of animals and computes allocation and patrolling strategies with drones and patrols using Double Deep Q Learning (DDQN).

Targets as a Part of Strategy

[74] takes a different approach to define the set of full strategy profiles as $X \times Y = (x^1, \dots, x^n, y^{n+1}, \dots, y^m)$. The defenders choose a strategy $x^\delta \in X$ and attackers choose a strategy $y^\alpha \in Y$ after observing x , implying targets as a part of the strategy. This approach relies on the definition that the defender strategy is a probability distribution over a set of actions that protect a limited number of targets and the attacker strategy is a probability distribution over a set of actions that attack a limited number of targets. However, targets are not necessarily associated with agent strategies as they are resources in SSGs.

[125] proposes the inducible Stackelberg equilibrium (ISE) based on the idea of an inducible target that serves as the lower bound of the defender's utility guarantees. Stackelberg security games and security games in general involve the idea of targets which are resources that the attackers tries to gain control from the defender and the defender protects from the attacker [158, 279, 316]. This motivates future work to develop a solution concept for stackelberg security games with the notion of targets in the domain of reinforcement learning as control of targets by the attacker implies that the game is lost while the opposite implies a defender's win.

Targets as a Separated Set

There are RL algorithms adapted for application in SSGs where the targets are considered separately as an input [154]. However these algorithms can only strategize for the defender's policies and will need to be extended to Multi-Agent RL for understanding optimal attacker policies also. [278] describes targets as a separated set, N , and an action s is an allocation of resources K over the set of targets N .

This form of definition most accurately reflects the proper definition of targets in SSG. However, targets as a separated set contradicts the existing modeling of SSG with MDP, POMDP, Dec-POMDP, and POSG where the tuple of sets do not leave room for definition of targets. If targets were to be modeled with a independent set, a new modeling of SSG is needed.

Challenge of No Explicit Definition of Targets

Existing literature in RL for SSGs has not modeled targets separately. Some papers modeling SSG as MDP [161, 214] or POMDP [7] lack an explicit definition of targets. [75, 302, 303] model SSGs as controllable Markov chains without considering target resources while adapting strategies against dynamic behavioral models of attackers. Given the significance of targets, not including its explicit definition introduces obscurity in modeling the payoff function which subsequently leads to ambiguity in the RL reward function.

For a sequential stochastic Stackelberg security game with one leader (defender) and one follower (attacker), [214] applies the idea of computing Markov equilibrium in linear time from Stackelberg equilibrium in double exponential time to a Markov Decision Process referred to as the Markov Perfect Stackelberg Equilibrium (MPSE). A model-free RL algorithm motivated from Expected Sarsa computes the optimal MPSE strategies for the defender and the attacker with the defender maximizing its own returns based on the knowledge that the attacker is strategizing to play the best response to the defender's strategy.

However, Strong Stackelberg Equilibrium (SSE), where the attacker (follower) breaks ties in the best response set by selecting the best response to the defender, is considered as the standard solution concept for Stackelberg security games [312] due to no guarantees about the existence of Weak Stackelberg Equilibrium (WSE) [78] which has not been addressed in [214].

Even the SSE may not be applicable for security games as the defender (leader) may not induce the attacker (follower) to select a preferred action with an infinitesimal change in their policy due to resource assignment constraints. [125] proposes the inducible Stackelberg equilibrium (ISE) based on the idea of an inducible target that serves as the lower bound of the defender's utility guarantees. Stackelberg security games and security games in general involve the idea of targets which are resources that the attackers tries to gain control from the defender and the defender protects from the attacker [158, 279, 316]. This motivates future work to develop a solution concept for stackelberg security games with the notion of targets in the domain of reinforcement learning. Control of targets by the attacker implies that the game is lost while the opposite implies a defender's win.

Existing research work applying repeated stackelberg security games with imperfect information in reinforcement learning [7] consider the targets to satisfy Markov and complex physical constraints where the games' strategies evolve around patrolling the targets. Likewise the learning algorithms to solve security games consider the targets to be nodes in the state space [161]. The algorithms proposed do not define targets explicitly which should be done for the solution of ISE to hold here as defined before. Targets can be moved from one state to another by attackers [316] demonstrating the control of agents on targets which is not captured in this modeling.

There are reinforcement learning algorithms adapted for application in Stackelberg security games where the targets are considered separately as an input [154]. However these algorithms can only strategize for the defender's policies and will need to be extended to Multi-Agent RL for understanding optimal attacker policies also.

8.4.6 Abstracting Targets

In practice, RL algorithms choose a policy that maximizes expected rewards over a highly randomized one. If the embedded SSGs can be modeled with targets set, resource constraints, and utilities, the game can be solved prior to learning policies to implement the winning strategy. Moreover, low-level decision control in MARL setting need not be confined to GT, SSGs can be solved with better time and sample complexity by abstracting the control problem.

8.4.7 Conclusion

Current literature in Multi-Agent Reinforcement Learning (MARL) to solve Stackelberg security games either ignores targets or considers them to be a state or strategy which does not reflect the solution concept of these games. Given the limitations, it may be beneficial to define the set of targets independent of existing parameters to learn better policies in RL. There are RL algorithms adapted for application in Stackelberg security games where the targets are considered separately as an input [154] to strategize for the defender's policies. These algorithms are being extended to MARL for understanding optimal attacker policies as well.

9.1 JAXMARL: Multi-Agent RL Environments and Algorithms in JAX

The nascent emergence of AI Agents has made benchmarks an important component, especially in Reinforcement Learning (RL), where CPU-bound environments often become a bottleneck for large-scale AI training. JAX’s hardware acceleration capabilities enable massively parallelized RL environment rollouts, mitigating this limitation. Multi-Agent Reinforcement Learning (MARL), a primary algorithmic paradigm to train AI Agents, has much to gain from this, where multiple agents interact at each step, putting more pressure on the computational overhead and sample complexity.

JAXMARL library enables training of Multi-Agent AI that combines ease-of-use in coding with parallel processing on GPUs, supporting widely used MARL environments that can be trained with the state-of-the-art baseline algorithms. This is one of the first open-source collaborative Multi-Agent AI code base that is aimed at creating major research infrastructure to unlock new research ideas, driven by the extremely fast training speeds which help to determine the success and/or quick adaptation of different Multi-Agent AI training methodologies.

JAXMARL offers up to a 12,500× speedup on common MARL environments and algo-

rithms compared to CPU-based implementations when considering wall clock time. I initiated and developed JAXMARL, contributing the vectorized environment rollouts and co-located agent updates on hardware accelerators with auto-vectorization, auto-differentiation, and Just-in-Time (JIT) compilation. This research library enables efficient and thorough evaluations, with the objective of alleviating the *evaluation crisis* in the field of AI Agents. The code base is open-source on Github.

9.2 Introduction

Benchmarks enable comparisons between different Multi-Agent Reinforcement Learning (MARL) algorithms by defining agentic problems, playing a pivotal role in the scientific process. Take Chess and Go as prime examples. These games serve as benchmarks, driving the development of MuZero [264]. In MARL decentralized StarCraft Micromanagement [104] and then the StarCraft Multi-Agent Challenge [SMAC, 261] led to the development of QMIX [249], a very popular MARL training algorithm.

A crucial bottleneck for simulation speed while training AI Agents, arises due to slow data transfer between the CPU and GPU that has to be done to evaluate the agents on GPU post simulation on CPU. Simulation speed plays a major role to ensure fast progress in Reinforcement Learning (RL). This is because a large number of environment interactions are needed to determine if an RL algorithm is improving the reward signal or learning how to make good decisions. This makes simulation speed a significant bottleneck. Non-stationarity and decentralized partial observability in MARL significantly worsen the sample complexity in MARL problems [30]. The latest acceleration in hardware and parallelization techniques for AI training are not provided

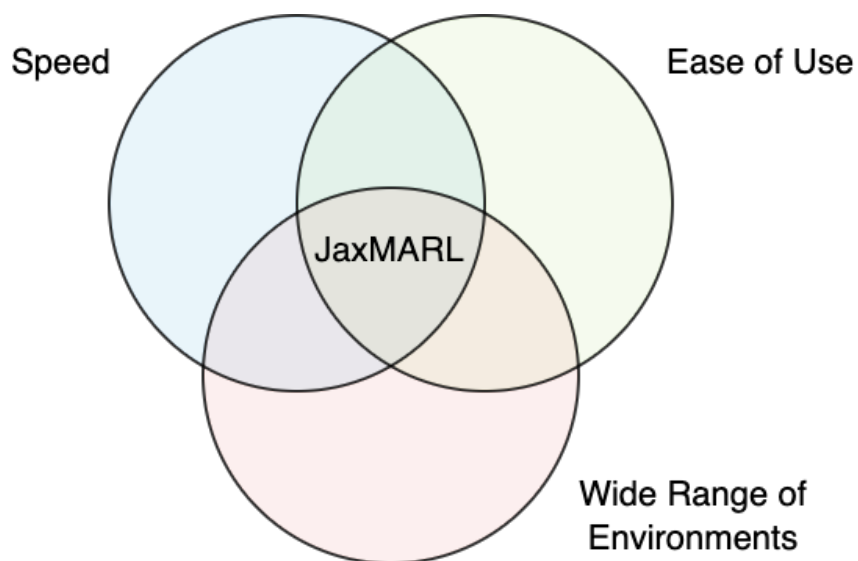


Figure 9.1: JAXMARL was built with the purpose of being an easy-to-use MARL library that does not compromise on efficiency and variety.

in Python, which limits the ability of machine learning researchers to train faster [269, 328]. The extremely efficient Hanabi library [137] created by the Meta-AI research team is written in C++, not Python, and as a result is less used by the MARL community than it otherwise might have been. JAX [40] enables direct use of Pythonic code with accelerated hardware (e.g. GPUs, TPUs), allowing for wider use of massively parallel RL environments and training algorithms.

JAX [40] provides function transformations that are composable. This allows for easy device parallelization, vectorization, autodiff and just-in-time (JIT) compilation with XLA [257] for optimization that works regardless of device. The environment rollouts and model training are both executed on a hardware accelerator (such as a GPU or TPU), using JAX. It removes the cost of data transfer between CPU and GPU devices which in return allows significant parallelization. Recently, the success of this end-to-end JAX-based approach has been observed in the PureJaxRL [192, 195] library. The model and the environment are both hosted on the GPU, yielding a $4,000\times$ speedup compared to a more “traditional” pipeline where the policy would be

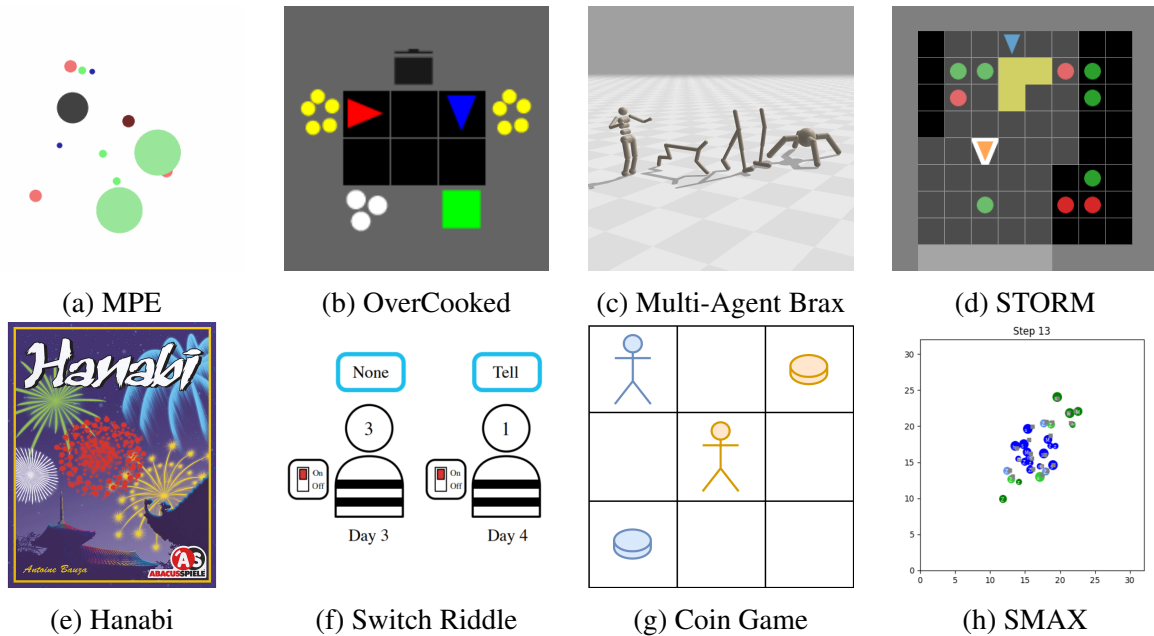


Figure 9.2: JAXMARL environments. Vectorized implementations of a wide range of environments are provided from different MARL settings.

evaluated on a GPU only after being simulated on the CPU.

These advances accelerate MARL and broader RL research significantly by making the testing and iteration of ideas with a real-world impact much faster. This is crucial for the development of decentralized AI Agents, which can be hosted on the edge with smartphones, Augmented Reality devices, and more. JAXMARL also lowers computational hurdles for extensive MARL research, meaning researchers can now use billions of frames and reach greater levels of performance than were feasible before.

Challenges with the evaluation standards across benchmarks in the MARL community are becoming prominent. MARL papers in particular usually only evaluate on a few domains. According to [118], only one evaluation environment is used in 50% of seventy-five recent MARL works. A further 30% of works only used two. While the extremely popular SMAC and

MPE [191] environments have different tasks or maps, the risk of biased comparisons and incorrect conclusions increases due to the lack of standardization. This leads to environment overfitting and unclear progress markers. Adding to the confusion is the lack of a unified codebase and a severe computational burden posed by comprehensive evaluations.

JAXMARL is presented in this section. It is a library that unifies JAX implementations of eight common MARL environments under one API for the first time, along with supporting JAX implementations of four state-of-the-art MARL algorithms. Similarly to PureJaxRL, this allows for end-to-end training pipelines built in JAX. As outlined in Figure 9.1, the JAXMARL library is presented with end-to-end hardware-accelerated training, simple Python implementations, and a broad range of MARL environments. Rapid evaluation of novel methods across a broad set of domains are allowed in JAXMARL which addresses MARL’s evaluation crisis. Specifically, JAXMARL achieves over $12,500\times$ speedup compared to “conventional” Multi-Agent AI approaches, including using PyTorch for major Multi-Agent AI environments.

The most popular environments from several MARL settings are included in the JAXMARL library. The Multi-Agent Particle Environments (MPE) [191], and Multi-Agent Brax (MABrax) are included under the centralized training with decentralized execution (CTDE) setting. Hanabi and Overcooked are included to cover zero-shot coordination (ZSC) and ad-hoc teamplay settings respectively. Also, CoinGame and Spatial-Temporal Representations of Matrix Games (STORM) are environments to support the setting of general-sum games. STORM provides a representation of matrix games as grid-world scenarios with temporally extended actions. These environments were not supported before by JAX, thereby allowing faster research with JAXMARL.

The JAX implementations of the four mostly commonly used MARL training algorithms, Independent PPO (IPPO) [85, 265], QMIX, VDN [294] and Independent Q -Learning

(IQL) [216], are provided in JAXMARL, allowing new techniques to be easily evaluated with baselines. The JAXMARL library also supports the popular MAPPO [341] algorithm which is helpful to finetune many Generative AI tasks because of its stability.

9.3 Background

9.3.1 Hardware Accelerated Environments

The usage of Python code with any hardware accelerator is allowed by JAX. It allows researchers to write hardware-accelerated code easily. Writing environment code in JAX has gained recent popularity, supported by useful research code like JAXMARL. This ensures two main advantages: firstly, JAX's `vmap` operation, which vectorizes functions across an input dimension, can ensure parallelization of environment runs, and secondly a JAX-coded environment allows it to be co-located with agent evaluations on the GPU, thereby removing the time taken to copy between CPU and GPU memory. Combining these factors increases training speed, with PureJaxRL [192] achieving a 4000x speedup over traditional training in single-agent settings.

9.4 JAXMARL

JAXMARL, an open-source Multi-Agent AI library, contains simple and accessible JAX implementations of popular MARL environments and algorithms, enabling significant acceleration and parallelization over existing baselines. A wide range of MARL environments and algorithms are provided in JAXMARL, one of the first open-source JAX-based Multi-Agent libraries, which is easy-to-use and accelerates training of Multi-Agent AI.

```

import jax
from JAXMARL import make

key = jax.random.PRNGKey(0)
key, key_reset, key_act, key_step = jax.random.split(key, 4)

#Initialize and reset the environment.
env = make('MPE_simple_world_comm_v3')
obs, state = env.reset(key_reset)

#Sample random actions.
key_act = jax.random.split(key_act, env.num_agents)
actions = {agent: env.action_space(agent).sample(key_act[i]) \
            for i, agent in enumerate(env.agents)}

#Perform the step transition.
obs, state, reward, done, infos = env.step(key_step, state, actions)

```

Figure 9.3: An example use of JAXMARL’s adaptable and intuitive API.

9.4.1 API

PettingZoo [299] and Gymnax has inspired the visualization interface of The interface of JAXMARL . This was designed to be a simple and easy-to-use interface for a wide-range of MARL problems. Figure 9.3 showcases an example to instantiate an environment from JAXMARL’s registry and to run one transition.

The `step` method has two additional inputs compared to PettingZoo’s, due to the support of pure scientific computing functions by JAX’s JII compilation. The `state` object stores the environment’s internal state which is updated with each call to `step`, before being passed to subsequent calls. Meanwhile, the `key_step` key is consumed by JAX functions that require stochasticity. This key is separated from the internal state to ensure clarity.

The remaining inputs and outputs are dictionaries keyed by agent names, similar to Petting-

Zoo, which allows differing action and observation spaces. However, the total number of agents in an environment cannot change during an episode as JAX’s JIT compilation requires arrays to have static shapes. Thus, the *agent iterator* in PettingZoo is not used. Instead, the maximum number of agents is limited during environment initialization. Any agents that terminate before the end of an episode pass dummy actions thereafter. The end of an episode is signaled using a special "`__all__`" key within `done` due to the possibility of asynchronous termination. The same dummy action approach is taken for environments where agents act asynchronously (e.g. turn-based games).

Strict registration of environments is provided as a feature with suffixed version numbers, to ensure clear and responsible coding, e.g. “MPE Simple Spread V3”. The version numbers match whenever JAXMARL environments correspond to existing CPU-based implementations.

9.4.2 Environments

JAXMARL contains diverse Agentic environments, all implemented in JAX. `SMAx 2s3z` and `27m30m`, SMAC-like environments for 2 stalkers vs 3 zealots and 27 marines vs 30 marines (stalkers, zealots, and marines are StarCraft characters with different capabilities) are implemented entirely in JAX. Full details are provided in [256]. This section provides details on how to implement the other provided environments in JAX.

Overcooked Overcooked is commonly used for assessing fully cooperative and fully observable Human-AI task performance. The goal is to quickly prepare and deliver soup by putting three onions in a pot, cooking the soup, and serving it into bowls. In order to maximize their shared reward signal, two agents, or *cooks*, must coordinate to effectively divide the tasks. The imple-

mentation follows the original from Overcooked-AI [53], including all five original layouts and a simple method for creating additional ones. Discussion on the limitations of the Overcooked-AI environment can be seen in [171].

Hanabi Hanabi is a fully cooperative partially observable multiplayer card game. The players can observe other players' cards but not their own. The team must play a series of cards in a specific order to win while sharing only a limited amount of information between players. Since reasoning about the beliefs and intentions of other agents is paramount to performance, it is a common benchmark for research on ZSC and ad-hoc teamplay. The Hanabi Learning Environment [27] has inspired the JAXMARL implementation which includes custom configurations for varying game settings, such as the number of colors/ranks, number of players, and number of hint tokens. While the Hanabi Learning Environment is written in C++ and split over dozens of files, the JAXMARL implementation is a single easy-to-read Python file, simplifying interactions with the library and running experiments.

Multi-Agent Particle Environments (MPE) A 2D world model with simple physics is featured in the multi-particle environments (MPE) in which particle agents can move, communicate, and interact with fixed landmarks. The format of the world and the agents' abilities varies in each environment, creating a diverse set of tasks that include both competitive and cooperative settings. All the MPE scenarios featured in the PettingZoo library are implemented, with the JAX implementation's transition logic matching that of the PyTorch based PettingZoo. Additionally a fully cooperative predator-prey variant of *simple tag* has been included as presented in [235]. The structured code allows for straightforward extensions, enabling further tasks to be added.

Multi-Agent Brax (MABrax) MABrax is derived from Multi-Agent MuJoCo [235], an extension of the MuJoCo Gym environment [300], commonly used for benchmarking continuous Multi-Agent robotic control. The underlying physics engine, provided in JAXMARL’s Brax[106], includes five of *Multi-Agent MuJoCo*’s Multi-Agent factorisation tasks in which each agent controls a subset of the joints and only observes the local state. The included tasks, illustrated in Figure 9.2, are: `ant_4x2`, `halfcheetah_6x1`, `hopper_3x1`, `humanoid_9|8`, and `walker2d_2x3`. The task descriptions mirror those from Gymnasium-Robotics [84].

Table 9.1: Benchmark results for JAX-based MARL environments (steps-per-second) when taking random actions. All environments are significantly faster than existing CPU implementations. [256]

Environment	Original, 1 Env	Jax, 1 Env	Jax, 100 Envs	Jax, 10k Envs	Maximum Speedup
MPE Simple Spread	8.34×10^4	5.48×10^3	5.24×10^5	3.99×10^7	4.78×10^2
MPE Simple Reference	1.46×10^5	5.24×10^3	4.85×10^5	3.35×10^7	2.29×10^2
Switch Riddle	2.69×10^4	6.24×10^3	7.92×10^5	6.68×10^7	2.48×10^3
Hanabi	2.10×10^3	1.36×10^3	1.05×10^5	5.02×10^6	2.39×10^3
Overcooked	1.91×10^3	3.59×10^3	3.04×10^5	1.69×10^7	8.85×10^3
MABrax Ant 4x2	1.77×10^3	2.70×10^2	1.81×10^4	7.62×10^5	4.31×10^2
Starcraft 2s3z	8.31×10^1	5.37×10^2	4.53×10^4	2.71×10^6	3.26×10^4
Starcraft 27m vs 30m	2.73×10^1	1.45×10^2	1.12×10^4	1.90×10^5	6.96×10^3
STORM	–	2.48×10^3	1.75×10^5	1.46×10^7	–
Coin Game	1.97×10^4	4.67×10^3	4.06×10^5	4.03×10^7	2.05×10^3

Coin Game Coin Game, a two-player grid-world environment, emulates social dilemmas, e.g. the iterated prisoner’s dilemma [280]. It expands on simpler general-sum social dilemmas by adding a high-dimensional state. Two players, ‘red’ and ‘blue’ move in a grid world and are each awarded 1 point for collecting any coin. The ‘red’ player loses 2 points if ‘blue’ collects a red coin and vice versa. Thus, the expected reward of both players is 0 if they ignore the coin color while collecting them. Further details are provided in Appendix 9.9.1.

Spatial-Temporal Representations of Matrix Games (STORM) The “in the Matrix” games in Melting Pot 2.0 [4] environment suite inspire the STORM [156] environment which expands on matrix games by representing them as grid-world scenarios. Resources are collected by agents which determine their strategy during interactions. Training of these strategies follow a pre-specified payoff matrix reward. This allows for the embedding of fully cooperative, competitive or general-sum games, such as the prisoner’s dilemma [280]. Thus, STORM can be used for studying paradigms such as *opponent shaping*, where agents act with the intent to change the learning dynamics of other agents’ that lead to more prosocial outcomes [103, 156, 194, 354]. A variety of new challenges such as partial observability, multi-step agent interactions, temporally-extended actions, and longer time horizons are provided in the grid-world setting in comparison to the Coin game or matrix games. STORM also includes stochastic features Unlike the “in the Matrix” games from Melting Pot, increasing the difficulty [93]. A further environment specification is provided in Appendix 9.9.2.

Switch Riddle Initially motivated to demonstrate the Differentiable Inter-Agent Learning algorithm [102], Switch Riddle serves as debugging tool and is thereby provided as a simple cooperative communication environment. n prisoners held by a warden can secure their release by collectively ensuring that each has passed through a room with a light bulb and a switch. Each day, a prisoner is selected at random to enter this room. They have three choices: do nothing, signal to the next prisoner by toggling the light, or inform the warden that they think all prisoners have been in the room. The game ends the moment when a prisoner informs the warden or the initialized maximum time steps elapses. The rewards are +1 if the prisoner informs the warden, with all prisoners being in the room, -1 if the prisoner informs the warden before all prisoners

have taken their turn, and 0 otherwise, including when the maximum time steps are reached. This implementation has been benchmarked from [350].

In this scenario, there are n prisoners held by a warden. To secure their release, the prisoners must collectively ensure that each has been in a room with a light bulb and switch. Each day, a prisoner is chosen at random to enter this room. They have three choices: do nothing, signal to the next prisoner by toggling the light, or inform the warden about their collective history.

9.4.3 Algorithms

In this subsection, JAX-based reimplementations of four well-known MARL baseline algorithms are provided. The primary objective of these baselines is to provide a structured framework for developing on-policy and off-policy MARL algorithms leveraging the advantages of these fast JAXMARL environments. All the training pipelines are fully compatible with JAX's JIT and VMAP functions, resulting in a significant acceleration of both the training and metric evaluation processes. This ensures parallelization of training with various seeds and hyperparameters on a single machine. The CleanRL philosophy has been followed in JAXMARL by providing clear, single-file implementations [141].

IPPO The PureJaxRL implementation of Independent PPO (IPPO) serves as the reference for the JAXMARL version. Homogenous agents share parameters. Two variants exist, one with a RNN and another with a feed-forward network.

Q-learning Methods The Q -learning Methods from PyMARL serve as the basis for the JAXMARL implementation of several baselines, including QMIX [248], Value Decomposition

Networks (VDN) [294] and Independent Q-Learning (IQL) [296]. This way, evaluations are consistent with published results and direct comparisons can be made between PyTorch implementations and the JAX version. Batched environments with aggregate trajectories are natively supported, simplifying parallelization as environments do not need to be run on separate threads and aggregated afterwards, as in PyMARL. A brief overview of the implemented baselines has been provided in the Appendix 9.9.

9.5 Results

The results aim to demonstrate the speed and correctness of environments and algorithms in JAXMARL. In several cases, minor changes to the environments signify that the environments do not exactly match the originals on a step-by-step level. Therefore, correctness for every environment is demonstrated in different ways which has been discussed separately. By combining this evidence, it can be seen that overall correct and extremely faster baselines are provided by the JAXMARL library on a wide range of sufficiently correct and easily-modifiable environments.

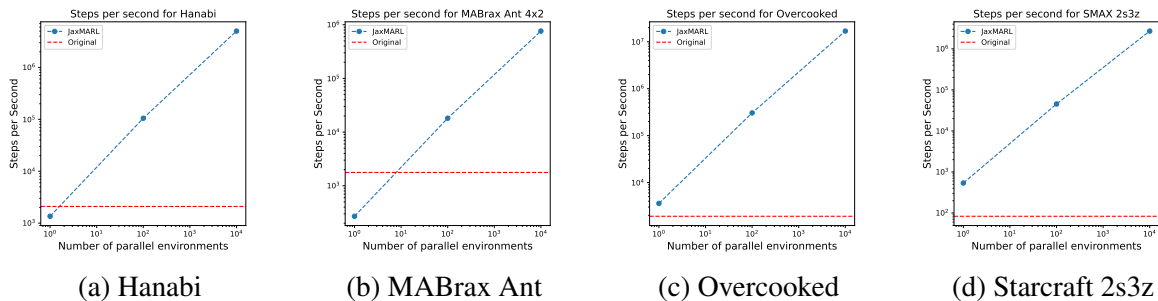


Figure 9.4: JAXMARL provides a considerable speedup compared to CPU-based environments as shown in these four examples [256].

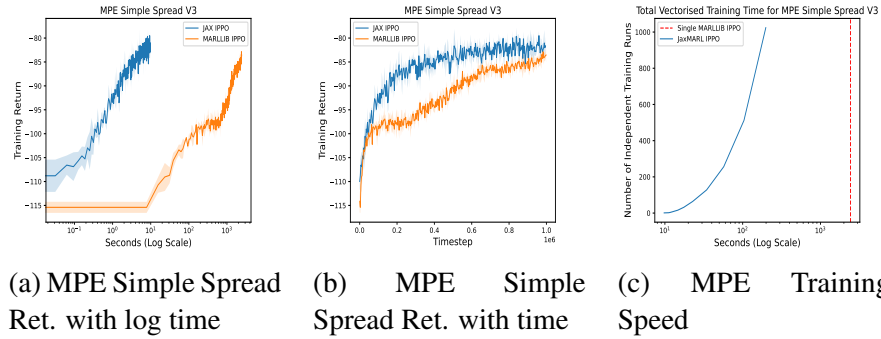


Figure 9.5: Speed and Performance (Return, abbreviated as Ret.) in JAXMARL compared to MARLLIB and PyMARL in MPE. Return results were averaged across 3 seeds. Performance results show 1 seed collected on the hardware described in Section 9.5.1.

9.5.1 Environment Speed

The performance of the JAX environments is measured in *steps per second* when using random actions and is compared with the original environments in Table 9.1 and Figure 9.4. All results are reported from one NVIDIA A100 GPU and AMD EPYC 7763 64-core processor. Environments roll-outs happen for 1000 sequential steps. Single environments have comparable performance to JAXMARL. But the ease of parallelization with JAX allows for more efficient and faster scaling compared to CPU-based environments. For example, MPE Simple Spread’s JAX implementation is $\sim 20x$ slower than the original when comparing a single environment, but even when only running 100 environments in parallel, the JAX environment is already over 6x faster. When considering 10000 environments, the JAX versions are much faster, achieving speedups of up to 8500x over the single-threaded environment (in the case of Overcooked). Running these many environments in parallel using CPU environments would have required a large CPU cluster and sophisticated communication mechanisms. This engineering is typically beyond the resources of academic labs and open-source researcher. Therefore JAXMARL helps to

unlock new research avenues for such institutions.

9.5.2 Algorithm Speed

An investigation of the speed of the trained IPPO algorithm’s JAX implementation is provided in Figure 9.5. By vectorizing over agents, a large population of agents can be trained in a fraction of the time it takes to train a single agent without hardware acceleration. For MPE, 1024 teams can be trained in 198.4 seconds, which is less than 0.2 seconds per team of agents. A single run of MARLLIB’s IPPO implementation on the same hardware takes around 2435.7 seconds on average. This represents a speedup of over $12,500\times$.

Significant speed advantages are also offered by the JAX-based Q -learning algorithms. In Figure 9.6a, training a single IQL, VDN, or QMIX policy in MPE takes ~ 130 seconds while the corresponding run with existing PyMARL library takes over an hour. Training 1024 QMIX learners in a batch requires 1670 seconds, translating to 1.6 seconds per learner, which indicates a $2700\times$ speedup. This speedup is not as large as for IPPO because Q -learning baselines are typically trained with fewer parallel environments. In the experiments, 8 parallel environments have been used for Q -learning compared to the 25 or 64 used for PPO. This is due to Q -learners benefiting more from a buffer with trajectories collected by different policies, resulting in a more frequent policy update, rather than collecting many trajectories with the same policy in parallel.

Speed-up results for the vectorized IPPO baseline with MAPPO [290] are compared for SMAX, the details of which are in [256]. 512 SMAX teams are trained on $2s3z$ in under 33 minutes, while it takes 44 hours on average for a single training run of PyTorch IPPO implementation. JAXMARL gives roughly a $40000\times$ speedup.

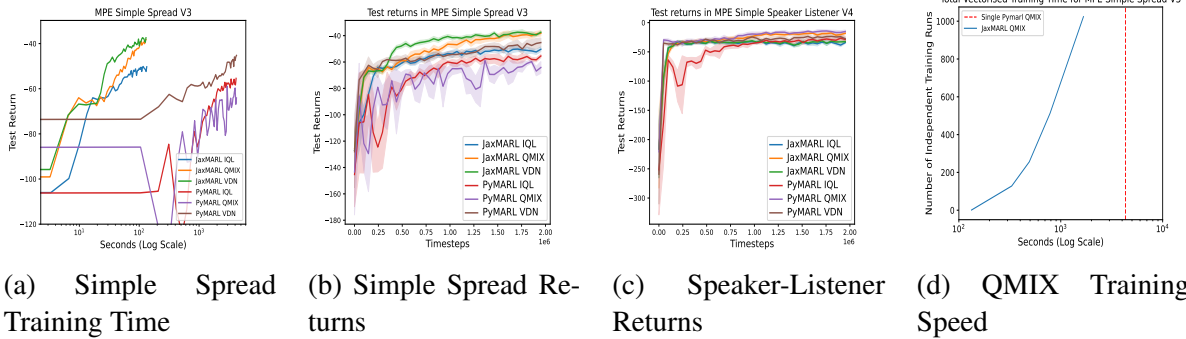


Figure 9.6: Performance and speed of JAXMARL Q -Learning baselines compared to PyMARL on MPE. These implementations match PyMARL’s returns, while being over 2000x faster to train

9.5.3 Algorithm Correctness

A comparative analysis of baselines from other libraries on the MPE Simple Spread and Simple Speaker Listener environments provide insights on the correctness of the algorithmic implementations in JAXMARL . For IPPO, the results consider the mean return across 3 seeds in Figure 9.5b. Results were collected on the same hardware as listed in Section 9.5.1. The IPPO implementation shows the same performance as MARLLIB and runs 250x quicker, taking only ten seconds to train.

For the Q -learning algorithms, the correctness is verified by comparing with PyMARL implementations of the same algorithms on the MPE Simple Spread and Simple Speaker Listener environments. IQL, VDN and QMIX obtain the same or better results than their PyMARL counterparts. The returns are from greedy policies and averaged across 8 runs. The hyperparameters used are from the PyMARL library.

9.5.4 Environment Correctness

MPE The MPE environment corresponds exactly to the PettingZoo implementation. This is validated for each environment using a uniform-random policy on 1000 rollouts. It ensures that all observations and rewards are within a tolerance of 1×10^{-4} at each transition. This tolerance accounts for non-determinism due to running floating point computation on the GPU. The performance of IPPO in Figure 9.5b and the Q -learning algorithms in Figures 9.6b and 9.6c respectively show this correspondence. The performance of these algorithms is inline with existing baselines [341]. Appendix 9.11.2 provides detailed training performance of IQL on the remaining MPE environments.

Overcooked The transition dynamics of the Overcooked implementation match those of the Overcooked-AI implementation. This is demonstrated by training an IPPO policy on the JAXMARL implementation and evaluating the policy on both the Overcooked implementation and the original at regular intervals. Results are illustrated in Figure 9.7a and performance is similar, demonstrating their equivalence.

MABrax As Brax differs subtly from MuJoCo, MABrax does not correspond to MAMuJoCo but the learning dynamics are qualitatively similar. To demonstrate this, mean training return is reported across 10 seeds for IPPO on `ant_4x2` in Figure 9.7b, and the results are in line with the performance of TRPO reported in [166]. The performance of IPPO on HalfCheetah and Walker is reported in Appendix 9.11.1, the results are also in line with TRPO.

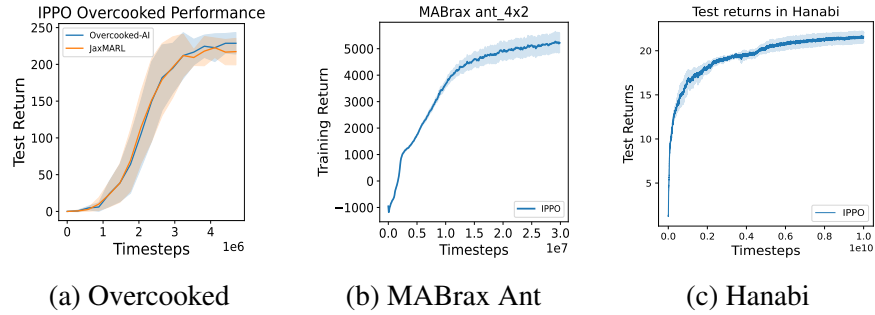


Figure 9.7: JAXMARL IPPO baseline results. These results correspond to similar baselines and therefore demonstrate the correctness of JAXMARL implementations.

Table 9.2: Recommended Minimal Environment Evaluations for different research settings

Setting	Recommended Environments
CTDE	SMAX (all scenarios), Hanabi (2-5 players), Overcooked
Zero-shot Coordination	Hanabi (2 players), Overcooked (5 basic scenarios)
General-Sum	STORM (iterated prisoner’s dilemma), STORM (matching pennies)
Cooperative Continuous Actions	MABrax

Hanabi The JAXMARL implementation of Hanabi does not correspond exactly to the Hanabi Learning Environment with a subtle difference in the observation space. The reasoning is provided in Appendix 9.9.3. To demonstrate qualitative similarity, IPPO is trained on Hanabi in self-play with 2 players, with the mean test return across 3 seeds reported in Figure 9.7c.

STORM, Coin Game & Switch Riddle STORM differs from Melting Pot 2.0 significantly, making direct comparisons challenging, with differences discussed in Appendix 9.9.2. Furthermore, STORM and Coin Game are general-sum games, so the environment returns of IPPO in self-play would not be a good indicator of performance. Switch Riddle is a simple diagnostic environment, however, it is not used for thorough evaluations.

9.6 Evaluation Recommendations

Existing research [118] highlights different evaluation practices between MARL algorithm research. The four main research areas that can benefit from the JAXMARL library are

1. cooperative centralized training with decentralized execution (CTDE) [102],
2. zero-shot coordination [138],
3. general-sum games,
4. and cooperative continuous action methods.

To aid comparisons between methods, it is recommended to use standard *minimal* sets of evaluation environments for each of these settings in Table 9.2. It’s important to note that these are *minimal* and it is encouraged to evaluate as broadly as possible. For example, in the zero-shot coordination setting, all methods should be able to evaluate on Hanabi and Overcooked. For some settings, such as continuous action environments and general-sum games, there is only one difficult environment. Further development of JAX-based environments in these settings can help to improve the quality of evaluation.

9.7 Related Work

Several open-source libraries exist for both MARL algorithms and environments. The popular library PyMARL [261] provides PyTorch implementations of QMIX, VDN and IQL and integrates easily with SMAC. E-PyMARL [234] extends this by adding the actor-critic algorithms MADDPG [191], MAA2C [215], IA2C [215], and MAPPO, and supports the SMAC, Gym [42],

Robot Warehouse [71], Level-Based Foraging [71], and MPE environments. Recently released MARLLib [140] is instead based on the open-source RL library RLLib [180] and combines a wide range of competitive, cooperative and mixed environments with a broad set of baseline algorithms. Meanwhile, MALib [359] focuses on population-based MARL across a wide range of environments. However, none of these frameworks feature hardware-accelerated environments and thus lack the associated performance benefits.

There has also been a recent proliferation of hardware-accelerated and JAX-based RL environments. Isaac gym [201] provides a GPU-accelerated simulator for a range of robotics platforms and CuLE [81] is a CUDA reimplementaion of the Atari Learning Environment [28]. Both of these environments are GPU-specific and cannot be extended to other hardware accelerators. Madrona [269] is an extensible game-engine written in C++ that allows for GPU acceleration and parallelization across environments. However, it requires environment code to be written in C++, limiting its accessibility. VMAS [32] provides a vectorized 2D physics engine written in PyTorch and a set of challenging multi-robot scenarios, including those from the MPE environment. For RL environments implemented in JAX, Jumanji [36] features mostly single-agent environments with a strong focus on combinatorial problems. The authors also provide an actor-critic baseline in addition to random actions. PGX [165] includes several board-game environments written in JAX. Gymnax [170] provides JAX implementations of the BSuite [231], classic continuous control, MinAtar [340] and other assorted environments. Gymnax’s sister-library, gymnax-baselines, provides PPO and ES baselines. Further extensions to Gymnax [193] also include POPGym environments [219]. Brax [106] reimplements the MuJoCo simulator in JAX and also provides a PPO implementation as a baseline. Jax-LOB [108] implements a vectorized limit order book as an RL environment that runs on the accelerator. Perhaps the most similar to the JAXMARL work

is Mava [241], which provides a MAPPO baseline, as well as integration with the Robot Warehouse environment. However, none of these libraries combine a range of JAX-based MARL environments with both value-based and actor-critic baselines. Broadly, no other work provides implementations of a wide range of hardware-accelerated MARL environments, while also implementing value-based and actor-critic baselines.

9.8 Conclusion

Hardware acceleration offers important opportunities for MARL research by lowering computational barriers, increasing the speed at which ideas can be iterated, and allowing for more thorough evaluation. JAXMARL, has been presented as an open-source library of popular MARL environments and baseline algorithms implemented in JAX. Ease of use is combined with hardware accelerator enabled efficiency to give significant speed-ups compared to traditional CPU-based implementations. Furthermore, by bringing together a wide range of MARL environments under one codebase, there is a potential to help alleviate issues with MARL’s evaluation standards. JAXMARL will help advance MARL by improving the ability of academic labs to conduct research with thorough, fast, and effective evaluations.

9.9 Appendix A: Further Details on Multi-Agent Environments

9.9.1 Coin Game

Two agents, ‘red’ and ‘blue’, move in a wrap-around grid and collect red and blue colored coins. An agent receives a reward of 1 when an agent collects any coin. However, ‘blue’ receives a reward of -2 when ‘red’ collects a blue coin and vice versa. Once a coin is collected, a new coin of the same color appears at a random location within the grid. If a coin is collected by both agents simultaneously, the coin is duplicated and both agents collect it. Episodes are of a set length.

9.9.2 Spatial-Temporal Representations of Matrix Games (STORM)

Directional agents are featured in this environment within an 8x8 grid-world with a restricted field of view. Agents cannot move backwards or share the same location. Collisions are resolved by either giving priority to the stationary agent or randomly if both are moving. Two unique resources: *cooperate* and *defect* coins are collected by agents. Once an agent picks up any coin, the agent’s color shifts, indicating its readiness to interact. The agents can then release an *interact* beam directly ahead; when this beam intersects with another ready agent, both are rewarded based on the specific matrix game payoff matrix. The agents’ coin collections determine their strategies. For instance, if an agent has 1 *cooperate* coin and 3 *defect* coins, there’s a 25% likelihood of the agent choosing to cooperate. After an interaction, the two agents in-

volved are frozen for five steps, revealing their coin collections to surrounding agents. After five steps, they respawn in a new location, with their coin count set back to zero. Once an episode concludes, the coin placements are shuffled. This grid-based approach to matrix games can be adapted for n-player versions. While STORM is inspired by MeltingPot 2.0, there are noteworthy differences:

- Meltingpot uses pixel-based observations while JAXMARL allows for direct grid access.
- Meltingpot’s grid size is typically 23x15, while ours is 8x8.
- Meltingpot features walls within its layout, ours does not.
- The JAXMARL environment introduces stochasticity by shuffling the coin placements, which remain static in Meltingpot.
- The agents begin with an empty coin inventory, making it easier for them to adopt pure cooperate or defect tactics, unlike in Meltingpot where they start with one of each coin.
- Existing MeltingPot is implemented in Lua [145] while we support it in JAX.

The coin shuffling is considered to be especially crucial because even large environments representing POMDPs, such as SMAC, can be solved without the need for memory if they lack sufficient randomness [93].

9.9.3 Hanabi

There are a few differences between the Hanabi implementation and the original Hanabi Learning Environment (HLE). The most notable of these is how card knowledge information is

represented in the agents' observation. In the HLE, card knowledge is observed as a color/rank if there has been an explicit hint about a given card. Implicit card knowledge is represented as a separate feature with possible colors/ranks if there has not been an explicit hint that indicates a given card is not that color/rank. We, on the other hand, combine implicit and explicit card knowledge, by only maintaining a representation of implicit card knowledge, which reduces to explicit card knowledge in the event an explicit hint is given about a card. This is because all possible colors/ranks are represented as 1s, whilst all ruled out colors/ranks are represented as 0s. By giving an explicit hint, all but one color/rank are ruled out, leaving a one-hot encoding of the explicit card knowledge. The card knowledge is implemented this way as knowledge updates are implemented via tensor calculus using JAX Numpy arrays of fixed shape and data type.

9.10 Appendix B: Value-Based MARL Methods and Implementation details

Key features of the JAXMARL framework include parameter sharing, a recurrent neural network (RNN) for agents, an epsilon-greedy exploration strategy with linear decay, a uniform experience replay buffer, and the incorporation of Double Deep Q -Learning (DDQN) [308] techniques to enhance training stability.

Unlike PyMARL, the Adam optimizer is used as the default optimization algorithm. Below is an introduction to common value-based MARL methods.

IQL (Independent Q -Learners) is a straightforward adaptation of Deep Q -Learning to multi-agent scenarios. It features multiple Q -Learner agents that operate independently, opti-

mizing their individual returns. This approach follows a decentralized learning and decentralized execution pipeline.

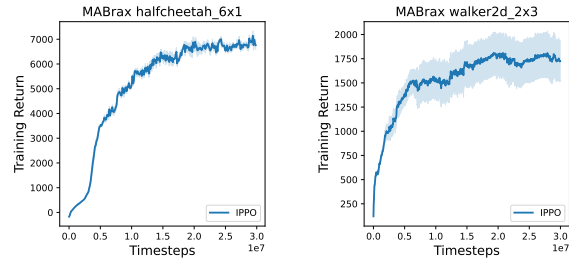
VDN (Value Decomposition Networks) extends Q -Learning to multi-agent scenarios with a centralized-learning-decentralized-execution framework. Individual agents approximate their own action’s Q -Value, which is then summed during training to compute a joint Q_{tot} for the global state-action pair. Back-propagation of the global DDQN loss in respect to a global team reward optimizes the factorization of the joint Q -Value.

QMIX improves upon VDN by relaxing the full factorization requirement. It ensures that a global *argmax* operation on the total Q -Value (Q_{tot}) is equivalent to individual *argmax* operations on each agent’s Q -Value. This is achieved using a feed-forward neural network as the mixing network, which combines agent network outputs to produce Q_{tot} values. The global DDQN loss is computed using a single shared reward function and is back-propagated through the mixer network to the agents’ parameters. Hypernetworks generate the mixing network’s weights and biases, ensuring non-negativity using an absolute activation function. These hypernetworks are two-layered multi-layer perceptrons with ReLU non-linearity.

9.11 Appendix C: Training Results

9.11.1 MABrax

The performance of IPPO on HalfCheetah and Walker is reported in Figure 9.8, with hyperparameters reported in Table 9.3.



(a) Speaker-Listener Returns (b) QMIX Training Speed

Figure 9.8: Performance of IPPO on MABrax Tasks [256]

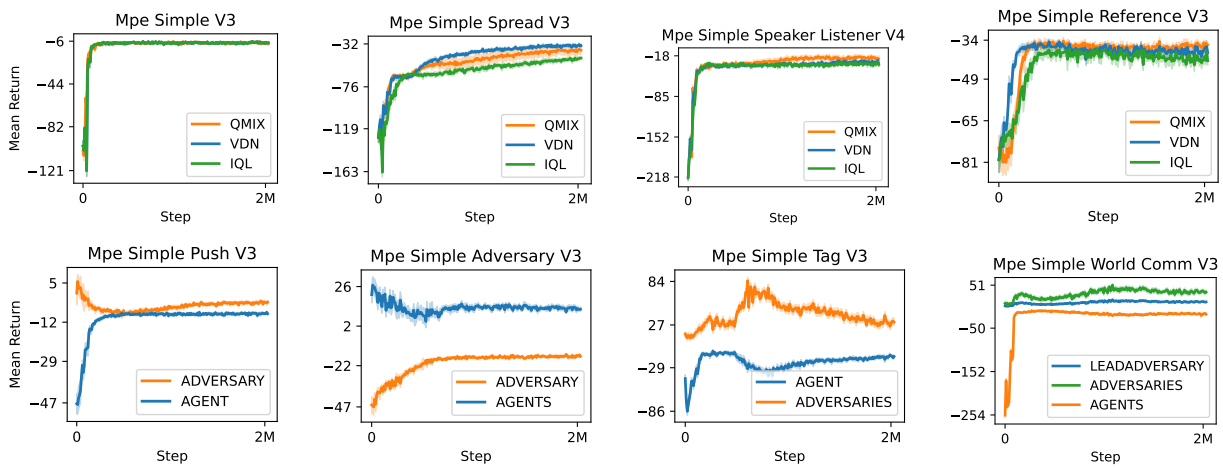


Figure 9.9: Q -Learning Baselines in all MPE scenarios. Where no algorithm names are given, the results represent IQL. [256]

9.11.2 MPE

Performance of Q -Learning baselines in all the MPE scenarios are reported in Figure 9.9. The upper row represents cooperative scenarios, with results for all the Q -learning baselines reported. The bottom row refers to competitive scenarios, and results for IQL are divided by agent types. Hyperparameters are given in Table 9.7

9.12 Appendix D: Hyperparameters

Value	Ant	HalfCheetah	Walker
VF_COEF	4.5	0.14	1.9
ENT_COEF	2×10^{-6}	4.5×10^{-3}	1×10^{-3}
LR	1×10^{-3}	6×10^{-4}	7×10^{-3}
NUM_ENVS	64	–	–
NUM_STEPS	300	–	–
TOTAL_TIMESTEPS	1×10^8	–	–
NUM_MINIBATCHES	4	–	–
GAMMA	0.99	–	–
GAE_LAMBDA	1.0	–	–
CLIP_EPS	0.2	–	–
MAX_GRAD_NORM	0.5	–	–
ACTIVATION	tanh	–	–
ANNEAL_LR	True	–	–

Table 9.3: MABrax Hyperparameters, where – indicates repeated parameters [256]

Hyperparameter	Value
LR	0.0005
NUM_ENVS	25
NUM_STEPS	128
TOTAL_TIMESTEPS	1×10^6
UPDATE_EPOCHS	5
NUM_MINIBATCHES	2
GAMMA	0.99
GAE_LAMBDA	1.0
CLIP_EPS	0.3
ENT_COEF	0.01
VF_COEF	1.0
MAX_GRAD_NORM	0.5
ACTIVATION	tanh
ANNEAL_LR	True

Table 9.4: Hyperparameters for MPE IPPO

Hyperparameter	Value
LR	5×10^{-4}
NUM_ENVS	1024
NUM_STEPS	128
TOTAL_TIMESTEPS	1×10^{10}
UPDATE_EPOCHS	4
NUM_MINIBATCHES	4
GAMMA	0.99
GAE_LAMBDA	0.95
CLIP_EPS	0.2
ENT_COEF	0.01
VF_COEF	0.5
MAX_GRAD_NORM	0.5
ACTIVATION	relu
ANNEAL_LR	True
NUM_FC_LAYERS	2
LAYER_WIDTH	512

Table 9.5: Hyperparameters for Hanabi IPPO [256]

Hyperparameter	Value
LR	2.5×10^{-4}
NUM_ENVS	16
NUM_STEPS	128
TOTAL_TIMESTEPS	5×10^6
UPDATE_EPOCHS	4
NUM_MINIBATCHES	4
GAMMA	0.99
GAE_LAMBDA	0.95
CLIP_EPS	0.2
ENT_COEF	0.01
VF_COEF	0.5
MAX_GRAD_NORM	0.5
ACTIVATION	tanh
ANNEAL_LR	True
NUM_EVALS	16

Table 9.6: Hyperparameters for Overcooked IPPO

Hyperparameter	Value
NUM_ENVS	8
NUM_STEPS	25
BUFFER_SIZE	5000
BUFFER_BATCH_SIZE	32
TOTAL_TIMESTEPS	2×10^6
AGENT_HIDDEN_DIM	64
AGENT_INIT_SCALE	2.0
EPSILON_START	1.0
EPSILON_FINISH	0.05
EPSILON_ANNEAL_TIME	100000
MIXER_EMBEDDING_DIM*	32
MIXER_HYPERNET_HIDDEN_DIM*	64
MIXER_INIT_SCALE*	0.00001
MAX_GRAD_NORM	25
TARGET_UPDATE_INTERVAL	200
LR	0.005
EPS_ADAM	0.001
WEIGHT_DECAY_ADAM	0.00001
GAMMA	0.9
NUM_TEST_EPISODES	32
TEST_INTERVAL	50000

Table 9.7: Hyperparameters for MPE Q-Learning Algorithms [256]

(* Parameters specific to QMix.)

Bibliography

- [1] Nicola K. Abram, Erik Meijaard, Kerrie A. Wilson, Jacqueline T. Davis, Jessie A. Wells, Marc Ancrenaz, Sugeng Budiharta, Alexandra Durrant, Afif Fakhruzzi, Rebecca K. Runting, David Gaveau, and Kerrie Mengersen. 2017. Oil palm–community conflict mapping in Indonesia: A case for better community liaison in planning for development initiatives. *Applied Geography* 78 (2017), 33–44. <https://doi.org/10.1016/j.apgeog.2016.10.005>
- [2] Steven Abreu, Tiffany D. Do, Karan Ahuja, Eric J. Gonzalez, Lee Payne, Daniel McDuff, and Mar Gonzalez-Franco. 2024. PARSE-Ego4D: Personal Action Recommendation Suggestions for Egocentric Videos. arXiv:2407.09503 [cs.CV] <https://arxiv.org/abs/2407.09503>
- [3] Manoj Acharya, Kushal Kafle, and Christopher Kanan. 2019. TallyQA: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 8076–8084.
- [4] John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. 2022. Melting Pot 2.0. *arXiv preprint arXiv:2211.13746* (2022).

- [5] Mehmet Akturk. 2020. Diabetes Dataset. <https://www.kaggle.com/datasets/mathchi/diabetes-data-set>. [Accessed 04-10-2023].
- [6] Mustafa Hamid Al-Jumaili and Yasa Ekşioğlu Özok. 2024. New control model for autonomous vehicles using integration of Model Predictive and Stanley based controllers. *Scientific Reports* 14, 1 (2024), 19872.
- [7] Guillermo Alcantara-Jiménez and Julio B. Clempner. 2020. Repeated Stackelberg security games: Learning with incomplete state information. *Reliability Engineering & System Safety* 195 (2020), 106695. <https://doi.org/10.1016/j.res.2019.106695>
- [8] Satya Almasian, Dennis Aumiller, and Michael Gertz. 2021. BERT got a Date: Introducing Transformers to Temporal Tagging. *arXiv preprint arXiv:2109.14927* (2021). <https://arxiv.org/abs/2109.14927>
- [9] Rapoport Anatol. 1966. "A Taxonomy of 2×2 Games." *General Systems. Yearbook of the Society for General Systems Research* 11 (1966), 203–214.
- [10] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record* 28, 2 (1999), 49–60.
- [11] Robert J . Aumann. 1964. 28. *Mixed and Behavior Strategies in Infinite Extensive Games*. Princeton University Press, Princeton, 627–650. <https://doi.org/doi:10.1515/9781400882014-029>

- [12] Kemen G Austin, Amanda Schwantes, Yaofeng Gu, and Prasad S Kasibhatla. 2019. What causes deforestation in Indonesia? *Environmental Research Letters* 14, 2 (2019), 024007.
- [13] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073* (2022).
- [14] Saptarashmi Bandyopadhyay. 2019. Factored Neural Machine Translation at LoResMT 2019. In *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*. European Association for Machine Translation, Dublin, Ireland, 68–71. <https://aclanthology.org/W19-6811>
- [15] Saptarashmi Bandyopadhyay. 2020. Factored neural machine translation on low resource languages in the covid-19 crisis. (2020).
- [16] Saptarashmi Bandyopadhyay, Vibhu Agrawal, Sarah Savidge, Eric Krokos, and John P Dickerson. 2023. Goal-Conditioned Recommendations of AI Explanations. In *NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning*. <https://openreview.net/forum?id=wQuszp213U>
- [17] Saptarashmi Bandyopadhyay, Vikas Bahirwani, Lavisha Aggarwal, Bhanu Guda, Lin Li, and Andrea Colaco. 2025. YETI (YET to Intervene) Proactive Interventions by Multimodal AI Agents in Augmented Reality Tasks. *arXiv:2501.09355 [cs.AI]* <https://arxiv.org/abs/2501.09355>
- [18] Saptarashmi Bandyopadhyay, Mustafa Mert Çelikok, and Robert Loftin. 2025. On the

- Complexity of Learning to Cooperate in Populations of Socially Rational Agents. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems* (Detroit, MI, USA) (AAMAS '25). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 233–241.
- [19] Saptarashmi Bandyopadhyay, John (Jack) Cole, Tom Goldstein, and David Jacobs. 2025. Multimodal Agentic Model Predictive Control. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems* (Detroit, MI, USA) (AAMAS '25). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2844–2848.
- [20] Saptarashmi Bandyopadhyay, Vahid Daneshpajoo, Danish Ahmad, Jennifer Toth, Rebecca Bascom, and William E Higgins. 2021. Blood vessel segmentation in narrow band imaging bronchoscopic video. In *Medical Imaging 2021: Biomedical Applications in Molecular, Structural, and Functional Imaging*, Vol. 11600. SPIE, 415–423.
- [21] Saptarashmi Bandyopadhyay, Tasnim Kabir, Zizhen Lian, and Marine Carpuat. 2021. The University of Maryland, College Park Submission to Large-Scale Multilingual Shared Task at WMT 2021. In *Proceedings of the Sixth Conference on Machine Translation*. 383–386.
- [22] Saptarashmi Bandyopadhyay, Deepthi Raghunandan, Dhruva Sahrawat, and John Dickerson. 2022. Preventing Deforestation: Modeling and Prediction of Vulnerabilities in Forest Conservation. In *Annual AAI Workshop on AI to Accelerate Science and Engineering (AI2ASE) at Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence 2022*.

- [23] Saptarashmi Bandyopadhyay, Jason Xu, Neel Pawar, and David Touretzky. 2022. Interactive Visualizations of Word Embeddings for K-12 Students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 12713–12720.
- [24] Saptarashmi Bandyopadhyay and Tianyang Zhao. 2020. Natural language response generation from SQL with generalization and back-translation. In *Proceedings of the First Workshop on Interactive and Executable Semantic Parsing*. 46–49.
- [25] Saptarashmi Bandyopadhyay, Chenqi Zhu, Philip Daniel, Joshua Morrison, Ethan Shay, and John Dickerson. 2022. Targets in Reinforcement Learning to solve Stackelberg Security Games. *arXiv preprint arXiv:2211.17132* (2022).
- [26] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2018. Emergent Complexity via Multi-Agent Competition. *arXiv:1710.03748 [cs.AI]*
- [27] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. 2020. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence* 280 (2020), 103216.
- [28] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47 (jun 2013), 253–279.
- [29] Richard Bellman. 1957. A Markovian Decision Process. *Indiana Univ. Math. J.* 6 (1957), 679–684. Issue 4.

- [30] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.
- [31] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. 2000. The Complexity of Decentralized Control of Markov Decision Processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (Stanford, California) (UAI'00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 32–37.
- [32] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. 2022. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems (2022)*.
- [33] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. 2024. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726* (2024).
- [34] Avishkar Bhoopchand, Bethanie Brownfield, Adrian Collister, Agustin Dal Lago, Ashley Edwards, Richard Everett, Alexandre Fréchette, Yanko Gitahy Oliveira, Edward Hughes, Kory W Mathewson, et al. 2023. Learning few-shot imitation as cultural transmission. *Nature Communications* 14, 1 (2023), 7536.
- [35] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. 2024. pi0:

A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164* (2024).

- [36] Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Vincent Coyette, Paul Duckworth, Laurence I. Midgley, Tristan Kalloniatis, Sasha Abramowitz, Cemlyn N. Waters, Andries P. Smit, Nathan Grinsztajn, Ulrich A. Mbou Sob, Omayma Mahjoub, Elshadai Tegegn, Mohamed A. Mimouni, Raphael Boige, Ruan de Kock, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. 2023. Jumanji: a Diverse Suite of Scalable Reinforcement Learning Environments in JAX. *arXiv:2306.09884* [cs.LG] <https://arxiv.org/abs/2306.09884>
- [37] Tilman Börgers and Rajiv Sarin. 1997. Learning through reinforcement and replicator dynamics. *Journal of economic theory* 77, 1 (1997), 1–14.
- [38] Robert N. Boute, Joren Gijsbrechts, Willem van Jaarsveld, and Nathalie Vanvuchelen. 2022. Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research* (April 2022).
- [39] Craig Boutilier, Martin Mladenov, and Guy Tennenholtz. 2023. Modeling recommender ecosystems: Research challenges at the intersection of mechanism design, reinforcement learning and generative models. *arXiv preprint arXiv:2309.06375* (2023).
- [40] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs*. <http://github.com/google/jax>

- [41] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (Dallas, Texas, USA) (SIGMOD '00)*. Association for Computing Machinery, New York, NY, USA, 93–104. <https://doi.org/10.1145/342009.335388>
- [42] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540
- [43] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817* (2022).
- [44] William Brown, Jon Schneider, and Kiran Vodrahalli. 2024. Is Learning in Games Good for the Learners? *Advances in Neural Information Processing Systems* 36 (2024).
- [45] Yash Butala, Siddhant Garg, Pratyay Banerjee, and Amita Misra. 2024. ProMISe: A Proactive Multi-turn Dialogue Dataset for Information-seeking Intent Resolution. In *Findings of the Association for Computational Linguistics: EACL 2024*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, St. Julian’s, Malta, 1774–1789. <https://aclanthology.org/2024.findings-eacl.124>
- [46] Caterina Bérubé, Marcia Nißen, Rasita Vinay, Alexa Geiger, Tobias Budig, Aashish Bhandari, Catherine Rachel Pe Benito, Nathan Ibarcena, Olivia Pistolese, Pan Li, Abdullah Bin Sawad, Elgar Fleisch, Christoph Stettler, Bronwyn Hemsley, Shlomo Berkovsky, Tobias

- Kowatsch, and A. Baki Kocaballi. 2024. Proactive behavior in voice assistants: A systematic review and conceptual model. *Computers in Human Behavior Reports* 14 (2024), 100411. <https://doi.org/10.1016/j.chbr.2024.100411>
- [47] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11621–11631.
- [48] Tianhui Cai, Yifan Liu, Zewei Zhou, Haoxuan Ma, Seth Z Zhao, Zhiwen Wu, and Jiaqi Ma. 2024. Driving with Regulation: Interpretable Decision-Making for Autonomous Vehicles with Retrieval-Augmented Reasoning via LLM. *arXiv preprint arXiv:2410.04759* (2024).
- [49] Ozgun Caliskan-Demirag, Youhua (Frank) Chen, and Yi Yang. 2012. Ordering Policies for Periodic-Review Inventory Systems with Quantity-Dependent Fixed Costs. *Operations Research* 60, 4 (Aug. 2012), 785–796.
- [50] Jacob Cannell. 2022. Empowerment is (almost) All We Need. <https://www.lesswrong.com/posts/JPHeENwRyXn9YFmXc/empowerment-is-almost-all-we-need>. [Online; accessed 9-March-2023].
- [51] Zhihao Cao, Zidong Wang, Siwen Xie, Anji Liu, and Lifeng Fan. 2024. Smart Help: Strategic Opponent Modeling for Proactive and Adaptive Robot Assistance in Households. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18091–18101.
- [52] Marta Caro-Martínez, Guillermo Jiménez-Díaz, and Juan A Recio-García. 2021. Con-

- ceptual modeling of explainable recommender systems: an ontological formalization to guide their design and development. *Journal of Artificial Intelligence Research* 71 (2021), 557–589.
- [53] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems* 32 (2019).
- [54] Micah D Carroll, Anca Dragan, Stuart Russell, and Dylan Hadfield-Menell. 2022. Estimating and penalizing induced preference shifts in recommender systems. In *International Conference on Machine Learning*. PMLR, 2686–2708.
- [55] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. arXiv:2307.15217 [cs.AI]
- [56] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge university press.
- [57] Doran Chakraborty and Peter Stone. 2010. Convergence, targeted optimality, and safety in multiagent learning. In *Proceedings of the 27th International Conference on International*

- Conference on Machine Learning* (Haifa, Israel) (*ICML'10*). Omnipress, Madison, WI, USA, 191–198.
- [58] Kranti Chalamalasetti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2023. clembench: Using game play to evaluate chat-optimized language models as conversational agents. *arXiv preprint arXiv:2305.13455* (2023).
- [59] Lawrence Chan, Dylan Hadfield-Menell, Siddhartha Srinivasa, and Anca Dragan. 2019. The assistive multi-armed bandit. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 354–363.
- [60] Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. 2023. Generating Diverse Cooperative Agents by Learning Incompatible Policies. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=UkU05GOH7_6
- [61] Vaibhav Chaudhary, Rakhee Kulshrestha, and Srikanta Routroy. 2018. State-of-the-art literature review on inventory models for perishable products. *Journal of Advances in Management Research* 15, 3 (2018), 306–346.
- [62] Katherine Chen and Michael Bowling. 2012. Tractable Objectives for Robust Policy Optimization. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2012/file/ce5140df15d046a66883807d18d0264b-Paper.pdf>
- [63] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin,

- Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.
- [64] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.
- [65] Xinyi Chen, Angelica Chen, Dean Foster, and Elad Hazan. 2024. Playing Large Games with Oracles and AI Debate. arXiv:2312.04792 [cs.GT]
- [66] Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. 2021. A survey of deep reinforcement learning in recommender systems: A systematic review and future directions. *arXiv preprint arXiv:2109.03540* (2021).
- [67] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335* (2024).
- [68] Zhaoyang Cheng, Guanpu Chen, and Yiguang Hong. 2022. Single-Leader-Multiple-Followers Stackelberg Security Game With Hypergame Framework. *IEEE Transactions on Information Forensics and Security* 17 (2022), 954–969. <https://doi.org/10.1109/TIFS.2022.3155294>
- [69] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. 2010. BART: Bayesian additive regression trees. *The Annals of Applied Statistics* 4, 1 (Mar 2010). <https://doi.org/10.1214/09-aoas285>

- [70] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30 (2017).
- [71] Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [72] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).
- [73] Kamil Ciosek. 2022. Imitation Learning by Reinforcement Learning. In *International Conference on Learning Representations*.
- [74] Julio B Clempner. 2022. Learning machiavellian strategies for manipulation in Stackelberg security games. *Annals of Mathematics and Artificial Intelligence* 90, 4 (2022), 373–395.
- [75] Julio B. Clempner. 2022. Reveling misleading information for defenders and attackers in repeated Stackelberg Security Games. *Engineering Applications of Artificial Intelligence* 110 (2022), 104703. <https://doi.org/10.1016/j.engappai.2022.104703>
- [76] John D Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine. 2019. Meta-Learning Language-Guided Policy Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HkgSEnA5KQ>

- [77] Vincent Conitzer and Tuomas Sandholm. 2003. AWESOME: A General Multiagent Learning Algorithm that Converges in Self-Play and Learns a Best Response Against Stationary Opponents. arXiv:cs/0307002 [cs.GT]
- [78] Vincent Conitzer and Tuomas Sandholm. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*. 82–90.
- [79] Brandon Cui, Hengyuan Hu, Luis Pineda, and Jakob Foerster. 2021. K-level Reasoning for Zero-Shot Coordination in Hanabi. *Advances in Neural Information Processing Systems* 34 (2021), 8215–8228.
- [80] Brandon Cui, Andrei Lupu, Samuel Sokota, Hengyuan Hu, David J Wu, and Jakob Nicolaus Foerster. 2023. Adversarial Diversity in Hanabi. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=uLE3WF3-H_5
- [81] Steven Dalton and iuri frosio. 2020. Accelerating Reinforcement Learning through GPU Atari Emulation. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 19773–19782. <https://proceedings.neurips.cc/paper/2020/file/e4d78a6b4d93e1d79241f7b282fa3413-Paper.pdf>
- [82] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2018. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

- [83] Daniela Pucci de Farias and Nimrod Megiddo. 2003. How to combine expert (or novice) advice when actions impact the environment. In *Advances in Neural Information Processing Systems*, Vol. 17.
- [84] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. 2023. *Gymnasium Robotics*. <http://github.com/Farama-Foundation/Gymnasium-Robotics>
- [85] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? arXiv:2011.09533 [cs.AI]
- [86] Yang Deng, Lizi Liao, Zhonghua Zheng, Grace Hui Yang, and Tat-Seng Chua. 2024. Towards Human-centered Proactive Conversational Agents. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (Washington DC, USA) (SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 807–818. <https://doi.org/10.1145/3626772.3657843>
- [87] Yuan Deng, Jon Schneider, and Balasubramanian Sivan. 2019. Strategizing against no-regret learners. *Advances in neural information processing systems* 32 (2019).
- [88] Ernst D Dickmanns and Alfred Zapp. 1987. Autonomous high speed road vehicle guidance by computer vision. *IFAC Proceedings Volumes* 20, 5 (1987), 221–226.
- [89] Yuandong Ding, Mingxiao Feng, Guozi Liu, Wei Jiang, Chuheng Zhang, Li Zhao, Lei Song, Houqiang Li, Yan Jin, and Jiang Bian. 2022. Multi-Agent Reinforcement Learning with Shared Resources for Inventory Management. In *RL4RealLife Workshop*.

- [90] Mustafa Doga Dogan, Eric J Gonzalez, Karan Ahuja, Ruofei Du, Andrea Colaço, Johnny Lee, Mar Gonzalez-Franco, and David Kim. 2024. Augmented Object Intelligence with XR-Objects. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. ACM, 1–15. <https://doi.org/10.1145/3654777.3676379>
- [91] Sourav Dutta, Jesujoba Alabi, Saptarashmi Bandyopadhyay, Dana Ruitter, and Josef van Genabith. 2020. UdS-DFKI@WMT20: Unsupervised MT and Very Low Resource Supervised MT for German-Upper Sorbian. In *Proceedings of the Fifth Conference on Machine Translation*. Association for Computational Linguistics, Online, 1092–1098. <https://aclanthology.org/2020.wmt-1.129>
- [92] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. 2004. Least angle regression. (2004).
- [93] Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. 2022. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489* (2022).
- [94] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, Vol. 96. 226–231.
- [95] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. 2021. Large scale interactive motion

- forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9710–9719.
- [96] Fei Fang and Thanh H Nguyen. 2016. Green security games: Apply game theory to addressing green security challenges. *ACM SIGecom Exchanges* 15, 1 (2016), 78–83.
- [97] Fei Fang, Peter Stone, and Milind Tambe. 2015. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
- [98] Tom G Farr, Paul A Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, et al. 2007. The shuttle radar topography mission. *Reviews of geophysics* 45, 2 (2007).
- [99] Jun Feng, Heng Li, Minlie Huang, Shichen Liu, Wenwu Ou, Zhirong Wang, and Xiaoyan Zhu. 2018. Learning to Collaborate: Multi-Scenario Ranking via Multi-Agent Reinforcement Learning. arXiv:1809.06260 [cs.AI]
- [100] Jenny Rose Finkel, Trond Grenager, and Christopher D Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*. 363–370.
- [101] Alessandro Flaborea, Guido Maria D’Amely di Melendugno, Leonardo Plini, Luca Scofano, Edoardo De Matteis, Antonino Furnari, Giovanni Maria Farinella, and Fabio Galasso. 2024. PREGO: online mistake detection in PROcedural EGOcentric videos. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
18483–18492.

- [102] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf
- [103] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with Opponent-Learning Awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 122–130.
- [104] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 1146–1155.
- [105] Agostino Forestiero. 2017. Multi-agent recommendation system in Internet of Things. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 772–775.
- [106] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier

- Bachem. 2021. *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation*. <http://github.com/google/brax>
- [107] Yoav Freund and Robert E Schapire. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29, 1-2 (1999), 79–103.
- [108] Sascha Frey, Kang Li, Peer Nagy, Silvia Sapora, Chris Lu, Stefan Zohren, Jakob Foerster, and Anisoara Calinescu. 2023. JAX-LOB: A GPU-Accelerated limit order book simulator to unlock large scale reinforcement learning for trading. *arXiv preprint arXiv:2308.13289* (2023).
- [109] Rachel Sarah Friedman. 2020. Managing forests for community, conservation, and social equity: a case study of social forestry in Indonesia. (2020).
- [110] Yoichiro Fukuda. 1964. Optimal Policies for the Inventory Problem with Negotiable Lead-time. *Management Science* (July 1964).
- [111] David Gaveau, Lucas Santos, Bruno Locatelli, Mohammad Salim, Husnayaen Msi, Erik Meijaard, Charlie Heatubun, and Douglas Sheil. 2021. Forest loss in Indonesian New Guinea: trends, drivers, and outlook. (02 2021). <https://doi.org/10.1101/2021.02.13.431006>
- [112] Aref Gholami and Abolfazl Mirzazadeh. 2018. An inventory model with controllable lead time and ordering cost, log-normal-distributed demand, and gamma-distributed available capacity. *Cogent Business & Management* (Jan. 2018).
- [113] Shahrzad Gholami, Amulya Yadav, Long Tran-Thanh, Bistra Dilkina, and Milind Tambe. 2019. Don't Put All Your Strategies in One Basket: Playing Green Security Games

- with Imperfect Prior Knowledge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) (AAMAS '19). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 395–403.
- [114] Akash Ghosh, Arkadeep Acharya, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. Exploring the frontier of vision-language models: A survey of current methodologies and future directions. *arXiv preprint arXiv:2404.07214* (2024).
- [115] Ilaria Giannoccaro and Pierpaolo Pontrandolfo. 2002. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics* 78, 2 (July 2002), 153–161.
- [116] Jennifer Gielis, Ajay Shankar, and Amanda Prorok. 2022. A Critical Review of Communications in Multi-robot Systems. *Current Robotics Reports* 3 (Aug. 2022), 213–225.
- [117] Harry Giles and Lawrence Chan. 2020. Accounting for Human Learning when Inferring Human Preferences. *arXiv preprint arXiv:2011.05596* (2020).
- [118] Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. 2022. Towards a Standardised Performance Evaluation Protocol for Co-operative MARL. *arXiv preprint arXiv:2209.10485* (2022).
- [119] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal,

Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina González, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jáchym Kolář, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbeláez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. 2022. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18995–19012.

- [120] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, Eugene Byrne, Zach Chavis, Joya Chen, Feng Cheng, Fu-Jen Chu, Sean Crane, Avijit Dasgupta, Jing Dong, Maria Escobar, Cristhian Forigua, Abrham Gebreselasie, Sanjay Haresh, Jing Huang, Md Mohaiminul Islam, Suyog Jain, Rawal Khirodkar, Devansh Kukreja, Kevin J Liang, Jia-Wei Liu, Sagnik Majumder, Yongsen Mao, Miguel Martin, Effrosyni Mavroudi, Tushar Nagarajan, Francesco Ragusa, Santhosh Kumar Ramakrishnan, Luigi Seminara, Arjun Somayazulu, Yale Song, Shan Su, Zihui Xue, Edward Zhang,

Jinxu Zhang, Angela Castillo, Changan Chen, Xinzhu Fu, Ryosuke Furuta, Cristina Gonzalez, Prince Gupta, Jiabo Hu, Yifei Huang, Yiming Huang, Weslie Khoo, Anush Kumar, Robert Kuo, Sach Lakhavani, Miao Liu, Mi Luo, Zhengyi Luo, Brigid Meredith, Austin Miller, Oluwatumininu Oguntola, Xiaqing Pan, Penny Peng, Shraman Pramanick, Merey Ramazanova, Fiona Ryan, Wei Shan, Kiran Somasundaram, Chenan Song, Audrey Southerland, Masatoshi Tateno, Huiyu Wang, Yuchen Wang, Takuma Yagi, Mingfei Yan, Xitong Yang, Zecheng Yu, Shengxin Cindy Zha, Chen Zhao, Ziwei Zhao, Zhifan Zhu, Jeff Zhuo, Pablo Arbelaez, Gedas Bertasius, David Crandall, Dima Damen, Jakob Engel, Giovanni Maria Farinella, Antonino Furnari, Bernard Ghanem, Judy Hoffman, C. V. Jawahar, Richard Newcombe, Hyun Soo Park, James M. Rehg, Yoichi Sato, Manolis Savva, Jianbo Shi, Mike Zheng Shou, and Michael Wray. 2024. Ego-Exo4D: Understanding Skilled Human Activity from First- and Third-Person Perspectives. arXiv:2311.18259 [cs.CV] <https://arxiv.org/abs/2311.18259>

[121] Tao Gui, Peng Liu, Qi Zhang, Liang Zhu, Minlong Peng, Yunhua Zhou, and Xuanjing Huang. 2019. Mention Recommendation in Twitter with Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (Paris, France) (SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 535–544. <https://doi.org/10.1145/3331184.3331237>

[122] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. 2024. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information*

Processing Systems 36 (2024).

- [123] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long Text Generation via Adversarial Training with Leaked Information. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.
- [124] Qingyu Guo, Bo An, Yevgeniy Vorobeychik, Long Tran-Thanh, Jiarui Gan, and Chunyan Miao. 2016. Coalitional security games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. 159–167.
- [125] Qingyu Guo, Jiarui Gan, Fei Fang, Long Tran-Thanh, Milind Tambe, and Bo An. 2019. On the inducibility of stackelberg equilibrium for security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2020–2028.
- [126] Xudong Guo, Daming Shi, and Wenhui Fan. 2023. Scalable Communication for Multi-Agent Reinforcement Learning via Transformer-Based Email Mechanism. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*.
- [127] Muskan Gupta, Priyanka Gupta, Jyoti Narwariya, Lovekesh Vig, and Gautam Shroff. 2024. SCM4SR: Structural Causal Model-based Data Augmentation for Robust Session-based Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2609–2613.
- [128] Yassine Hachaichi, Yassine Chemingui, and Mariem Affes. 2020. A Policy Gradient Based Reinforcement Learning Method for Supply Chain Management. In *International Conference on Advanced Systems and Emergent Technologies*.

- [129] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2016. Cooperative inverse reinforcement learning. *Advances in neural information processing systems* 29 (2016).
- [130] Isha Hameed, Samuel Sharpe, Daniel Barcklow, Justin Au-Yeung, Sahil Verma, Jocelyn Huang, Brian Barr, and C. Bayan Bruss. 2022. BASED-XAI: Breaking Ablation Studies Down for Explainable Artificial Intelligence. arXiv:2207.05566 [cs.LG] <https://arxiv.org/abs/2207.05566>
- [131] James Hannan. 1957. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games* 3, 2 (1957), 97–140.
- [132] M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. A. Turubanova, A. Tyukavina, D. Thau, S. V. Stehman, S. J. Goetz, T. R. Loveland, A. Komareddy, A. Egorov, L. Chini, C. O. Justice, and J. R. G. Townshend. 2013. High-Resolution Global Maps of 21st-Century Forest Cover Change. *Science* 342, 6160 (2013), 850–853. <https://doi.org/10.1126/science.1244693>
arXiv:<https://science.sciencemag.org/content/342/6160/850.full.pdf>
- [133] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 720–730.
- [134] Mark K Ho, Michael Littman, James MacGlashan, Fiery Cushman, and Joseph L Auster-

- weil. 2016. Showing versus doing: Teaching by demonstration. *Advances in neural information processing systems* 29 (2016).
- [135] Josef Hofbauer and Karl Sigmund. 2003. Evolutionary game dynamics. *Bulletin of the American mathematical society* 40, 4 (2003), 479–519.
- [136] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. 2020. Language and Visual Entity Relationship Graph for Agent Navigation. *Advances in Neural Information Processing Systems* 33 (2020).
- [137] Hengyuan Hu and Jakob N Foerster. 2020. Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=B1xm3RVtwB>
- [138] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. 2020. “other-play” for zero-shot coordination. In *International Conference on Machine Learning*. PMLR, 4399–4410.
- [139] Hengyuan Hu and Dorsa Sadigh. 2023. Language Instructed Reinforcement Learning for Human-AI Coordination. In *International Conference on Machine Learning*.
- [140] Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Zhihui Li, Xiaodan Liang, Xiaojun Chang, and Yaodong Yang. 2022. MARLlib: Extending RLlib for Multi-agent Reinforcement Learning. *arXiv preprint arXiv:2210.13708* (2022).
- [141] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. 2022. CleanRL: High-quality Single-

- file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research* 23, 274 (2022), 1–18. <http://jmlr.org/papers/v23/21-1342.html>
- [142] Alihan Hüyük, Daniel Jarrett, and Mihaela van der Schaar. 2022. Inverse Contextual Bandits: Learning How Behavior Evolves over Time. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 9506–9524. <https://proceedings.mlr.press/v162/huyuk22a.html>
- [143] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847* (2019).
- [144] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A tractable decomposition for reinforcement learning with recommendation sets. (2019).
- [145] Roberto Ierusalimschy. 2006. *Programming in lua*. Roberto Ierusalimschy.
- [146] Jeremy Irvin, Hao Sheng, Neel Ramachandran, Sonja Johnson-Yu, Sharon Zhou, Kyle Story, Rose Rustowicz, Cooper Elsworth, Kemen Austin, and Andrew Y Ng. 2020. Forest-Net: Classifying Drivers of Deforestation in Indonesia using Deep Learning on Satellite Imagery. *arXiv preprint arXiv:2011.05479* (2020).
- [147] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Don-

- ahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).
- [148] Hong Jun Jeon, Smitha Milli, and Anca Dragan. 2020. Reward-rational (implicit) choice: A unifying formalism for reward learning. *Advances in Neural Information Processing Systems* 33 (2020), 4415–4426.
- [149] Ellis L. Johnson. 1968. On (s, S) Policies. *Management Science* (Sept. 1968).
- [150] Matthew Johnson, Fei Fang, and Milind Tambe. 2012. Patrol strategies to maximize pristine forest area. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 26.
- [151] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169* (2023).
- [152] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134. [https://doi.org/10.1016/S0004-3702\(98\)00023-X](https://doi.org/10.1016/S0004-3702(98)00023-X)
- [153] Debarun Kar. 2017. *When AI helps Wildlife Conservation: Learning Adversary Behaviors in Green Security Games*. Ph. D. Dissertation. University of Southern California.
- [154] Debarun Kar, Fei Fang, Francesco M Delle Fave, Nicole Sintov, Milind Tambe, and Arnaud Lyet. 2016. Comparing human behavior models in repeated Stackelberg security games: An extended study. *Artificial Intelligence* 240 (2016), 65–103.
- [155] Narayan Kayet, Khanindra Pathak, Subodh Kumar, CP Singh, VM Chowdary, Abhisek

- Chakrabarty, Nibedita Sinha, Ibrahim Shaik, and Amit Ghosh. 2021. Deforestation susceptibility assessment and prediction in hilltop mining-affected forest region. *Journal of Environmental Management* 289 (2021), 112504.
- [156] Akbir Khan, Newton Kwan, Timon Willi, Chris Lu, Andrea Tacchetti, and Jakob Nicolaus Foerster. 2022. Context and History Aware Other-Shaping. (2022).
- [157] Madhav Khirwar, Karthik S. Gurumoorthy, Ankit Ajit Jain, and Shantala Manchenahally. 2023. Cooperative Multi-Agent Reinforcement Learning for Inventory Management. <https://doi.org/10.48550/ARXIV.2304.08769>
- [158] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. 2009. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. 689–696.
- [159] Changyeon Kim, Younggyo Seo, Hao Liu, Lisa Lee, Jinwoo Shin, Honglak Lee, and Kimin Lee. 2023. Guide Your Agent with Adaptive Multimodal Rewards. In *Thirty-seventh Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=G8nal7MpIQ>
- [160] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246 [cs.RO] <https://arxiv.org/abs/2406.09246>

- [161] Richard Klíma, Karl Tuyls, and Frans A. Oliehoek. 2016. Markov Security Games : Learning in Spatial Security Problems.
- [162] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. 2005. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, Vol. 1. IEEE, 128–135.
- [163] Paul Knott, Micah Carroll, Sam Devlin, Kamil Ciosek, Katja Hofmann, A. D. Dragan, and Rohin Shah. 2021. Evaluating the Robustness of Collaborative Agents. <https://doi.org/10.48550/ARXIV.2101.05507>
- [164] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.
- [165] Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. 2023. Pgx: Hardware-accelerated Parallel Game Simulators for Reinforcement Learning. *arXiv preprint arXiv:2303.17503* (2023).
- [166] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2022. TRUST REGION POLICY OPTIMISATION IN MULTI-AGENT REINFORCEMENT LEARNING. In *Tenth International Conference on Learning Representations*.
- [167] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain

- Gelly. 2020. Google Research Football: A Novel Reinforcement Learning Environment. arXiv:1907.11180 [cs.LG]
- [168] Asad M Lak, Abdullah M Abunimer, Caroline MW Goedmakers, Linda S Aglio, Timothy R Smith, Melvin Makhni, Rania A Mekary, and Hasan A Zaidi. 2021. Single-versus Dual-Attending Surgeon Approach for Spine Deformity: A Systematic Review and Meta-Analysis. , 233–241 pages.
- [169] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. arXiv:1711.00832 [cs.AI]
- [170] Robert Tjarko Lange. 2022. *gymnax: A JAX-based Reinforcement Learning Environment Library*. <http://github.com/RobertTLange/gymnax>
- [171] Niklas Lauffer, Ameesh Shah, Micah Carroll, Michael D Dennis, and Stuart Russell. 2023. Who Needs to Know? Minimal Knowledge for Optimal Coordination. In *International Conference on Machine Learning*. PMLR, 18599–18613.
- [172] Angeliki Lazaridou and Marco Baroni. 2020. Emergent Multi-Agent Communication in the Deep Learning Era. *CoRR* abs/2006.02419 (2020).
- [173] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning*. PMLR, 1995–2003.
- [174] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash.

2023. RLAIIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. arXiv:2309.00267 [cs.CL]
- [175] Remi Leluc, Elie Kadoche, Antoine Bertoncello, and Sebastien Gourvenec. 2022. MARLIM: Multi-Agent Reinforcement Learning for Inventory Management. In *Reinforcement Learning for Real Life Workshop*.
- [176] Cheng Li, Mingyang Zhang, Qiaozhu Mei, Yaqing Wang, Spurthi Amba Hombaiah, Yi Liang, and Michael Bendersky. 2023. Teach LLMs to Personalize – An Approach inspired by Writing Education.
- [177] Junyi Li, Siqing Li, Wayne Xin Zhao, Gaole He, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2020. Knowledge-Enhanced Personalized Review Generation with Capsule Graph Neural Network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (Virtual Event, Ireland) (CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 735–744. <https://doi.org/10.1145/3340531.3411893>
- [178] Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023. GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation.
- [179] Junyi Li, Wayne Xin Zhao, Ji-Rong Wen, and Yang Song. 2019. Generating Long and Informative Reviews with Aspect-Aware Coarse-to-Fine Decoding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.
- [180] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg,

- Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International conference on machine learning*. PMLR, 3053–3062.
- [181] Guibiao Liao, Jiankun Li, and Xiaoqing Ye. 2024. VLM2Scene: Self-Supervised Image-Text-LiDAR Learning with Foundation Models for Autonomous Driving Scene Understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 3351–3359.
- [182] Lizi Liao, Grace Hui Yang, and Chirag Shah. 2023. Proactive Conversational Agents in the Post-ChatGPT World. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (Taipei, Taiwan) (SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 3452–3455. <https://doi.org/10.1145/3539618.3594250>
- [183] Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*. 157–163.
- [184] Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. 2023. Chain of Hindsight Aligns Language Models with Feedback. arXiv:2302.02676 [cs.LG]
- [185] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).
- [186] Robert Loftin, Saptarashmi Bandyopadhyay, and Mustafa Mert Çelikok. 2024. On the

Complexity of Learning to Cooperate with Populations of Socially Rational Agents.
arXiv:2407.00419 [cs.LG] <https://arxiv.org/abs/2407.00419>

- [187] Robert Loftin and Frans A Oliehoek. 2022. On the Impossibility of Learning to Cooperate with Adaptive Partner Strategies in Repeated Games. In *International Conference on Machine Learning*. PMLR, 14197–14209.
- [188] Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. 2016. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous agents and multi-agent systems* 30 (2016), 30–59.
- [189] Robert Loftin, Mustafa Mert Çelikok, and Frans A. Oliehoek. 2023. Towards a Unifying Model of Rationality in Multiagent Systems. arXiv:2305.18071 [cs.AI]
- [190] Keke Long, Haotian Shi, Jiayi Liu, and Xiaopeng Li. 2024. VLM-MPC: Vision Language Foundation Model (VLM)-Guided Model Predictive Controller (MPC) for Autonomous Driving. arXiv:2408.04821 [cs.RO] <https://arxiv.org/abs/2408.04821>
- [191] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (Long Beach, California, USA) (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6382–6393.
- [192] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and

- Jakob Foerster. 2022. Discovered policy optimisation. *Advances in Neural Information Processing Systems* 35 (2022), 16455–16468.
- [193] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. 2023. Structured State Space Models for In-Context Reinforcement Learning. *arXiv e-prints* (2023), arXiv–2303.
- [194] Christopher Lu, Timon Willi, Christian A Schroeder De Witt, and Jakob Foerster. 2022. Model-Free Opponent Shaping. In *International Conference on Machine Learning*. PMLR, 14398–14411.
- [195] Chris Lu, Timon Willi, Alistair Letcher, and Jakob Nicolaus Foerster. 2023. Adversarial cheap talk. In *International Conference on Machine Learning*. PMLR, 22917–22941.
- [196] Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2024. Proactive Agent: Shifting LLM Agents from Reactive Responses to Active Assistance. arXiv:2410.12361 [cs.AI] <https://arxiv.org/abs/2410.12361>
- [197] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).
- [198] Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. 2021. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*. PMLR, 7204–7213.

- [199] Zixian Ma, Rose Wang, Fei-Fei Li, Michael Bernstein, and Ranjay Krishna. 2022. ELIGN: Expectation Alignment as a Multi-Agent Intrinsic Reward. *Advances in Neural Information Processing Systems* 35 (2022), 8304–8317.
- [200] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) (*ICML'17*). JMLR.org, 2285–2294.
- [201] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning.
- [202] Alberto Marchesi, Gabriele Farina, Christian Kroer, Nicola Gatti, and Tuomas Sandholm. 2019. Quasi-perfect stackelberg equilibrium. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2117–2124.
- [203] Luca Marconi, R Matamoros Aragon, Francesco Epifania, et al. 2023. Discovering the unknown suggestion: a short review on explainability for recommender systems. In *CEUR Workshop Proceedings*, Vol. 3463. CEUR-WS, 1–15.
- [204] V. N. Marivate, G. Ssali, and T. Marwala. 2008. An intelligent Multi-Agent recommender system for human capacity building. In *MELECON 2008 - The 14th IEEE Mediterranean Electrotechnical Conference*. IEEE. <https://doi.org/10.1109/melcon.2008.4618553>

- [205] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. 1995. *Microeconomic theory*. Vol. 1. Oxford university press New York.
- [206] Michele Mazzamuto, Antonino Furnari, and Giovanni Maria Farinella. 2024. Eyes Wide Unshut: Unsupervised Mistake Detection in Egocentric Video by Detecting Unpredictable Gaze. *arXiv preprint arXiv:2406.08379* (2024).
- [207] Sara Marie Mc Carthy, Milind Tambe, Christopher Kiekintveld, Meredith Gore, and Alex Killion. 2016. Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.
- [208] Ian R. McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, Andrew Gritsevskiy, Daniel Wurgaft, Derik Kauffman, Gabriel Recchia, Jiacheng Liu, Joe Cavanagh, Max Weiss, Sicong Huang, The Floating Droid, Tom Tseng, Tomasz Korbak, Xudong Shen, Yuhui Zhang, Zhengping Zhou, Najoung Kim, Samuel R. Bowman, and Ethan Perez. 2023. Inverse Scaling: When Bigger Isn't Better. *arXiv:2306.09479 [cs.CL]*
- [209] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *arXiv:2312.02119 [cs.LG]*
- [210] Erik Miebling, Rahul Nair, Elizabeth Daly, Karthikeyan Natesan Ramamurthy, and Robert Redmond. 2024. Cookie consent has disparate impact on estimation accuracy. *Advances in Neural Information Processing Systems* 36 (2024).

- [211] Vivek Miglani, Aobo Yang, Aram H Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. 2023. Using captum to explain generative language models. *arXiv preprint arXiv:2312.05491* (2023).
- [212] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
- [213] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. 2022. A survey of ad hoc teamwork: Definitions, methods, and open problems. In *European Conference on Multiagent Systems*.
- [214] Rajesh K Mishra, Deepanshu Vasal, and Sriram Vishwanath. 2020. Model-free Reinforcement Learning for Stochastic Stackelberg Security Games. <https://doi.org/10.48550/ARXIV.2005.11853>
- [215] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.
- [216] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

- [217] Barnabé Monnot and Georgios Piliouras. 2017. Limits and limitations of no-regret learning in games. *The Knowledge Engineering Review* 32 (2017), e21.
- [218] Juan P Monzon, Maja A Slingerland, Suroso Rahutomo, Fahmuddin Agus, Thomas Oberthür, José F Andrade, Antoine Couédel, Juan I Rattalino Edreira, Willem Hekman, Rob van den Beuken, et al. 2021. Fostering a climate-smart intensification for oil palm. *Nature Sustainability* (2021), 1–7.
- [219] Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. 2023. POPGym: Benchmarking Partially Observable Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=chDrutUTs0K>
- [220] A Jorge Morais, Eugénio Oliveira, and Alípio Mário Jorge. 2012. A multi-agent recommender system. In *Distributed Computing and Artificial Intelligence: 9th International Conference*. Springer, 281–288.
- [221] Matthew Morris, Thomas D Barrett Barret, and Arnú Pretorius. 2022. Universally Expressive Communication in Multi-Agent Reinforcement Learning. *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)* (2022). <https://arxiv.org/abs/2206.06758>
- [222] Tong Mu, Stephan Zheng, and Alexander R Trott. 2022. Modeling Bounded Rationality in Multi-Agent Simulations Using Rationally Inattentive Reinforcement Learning. *Transactions on Machine Learning Research* (2022). <https://openreview.net/forum?id=DY1pMrmDkm>

- [223] Muhammad Firdaus Mujibuddin Syah Mustafa, Namasivayam Navaranjan, and Amer Demirovic. 2024. Food cold chain logistics and management: A review of current development and emerging trends. *Journal of Agriculture and Food Research* 18 (2024), 101343. <https://doi.org/10.1016/j.jafvr.2024.101343>
- [224] Roger B Myerson. 2013. *Game theory*. Harvard university press.
- [225] Steven Nahmias. 1982. Perishable Inventory Theory: A Review. *Operations research* 30 (08 1982), 680–708. <https://doi.org/10.1287/opre.30.4.680>
- [226] Suraj Nair. [n. d.]. When Security Games Go Green. https://www.cs.umd.edu/class/spring2018/cmsc828m/lecs/cmsc828m_s2018_lec9_suraj_nair.pdf. [Online; accessed 9-April-2021].
- [227] Jianmo Ni and Julian McAuley. 2018. Personalized Review Generation By Expanding Phrases and Attending on Aspect-Aware Representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.
- [228] Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. 2024. Screenagent: A vision language model-driven computer control agent. *arXiv preprint arXiv:2402.07945* (2024).
- [229] OpenAI. 2023. GPT-4 Technical Report. [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) [cs.CL]
- [230] Afshin Oroojlooy and Davood Hajinezhad. 2022. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence* (Oct. 2022).
- [231] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva,

- Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, et al. 2019. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568* (2019).
- [232] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.
- [233] R Kelley Pace and Ronald Barry. 1997. Sparse spatial autoregressions. *Statistics & Probability Letters* 33, 3 (1997), 291–297.
- [234] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. 2020. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869* (2020).
- [235] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems* 34 (2021), 12208–12221.
- [236] Hector D. Perez, Christian D. Hubbs, Can Li, and Ignacio E. Grossmann. 2021. Algorithmic Approaches to Inventory Management Optimization. *Processes* (Jan. 2021).
- [237] Leonardo Plini, Luca Scofano, Edoardo De Matteis, Guido Maria D’Amely di Melenugno, Alessandro Flaborea, Andrea Sanchietti, Giovanni Maria Farinella, Fabio Galasso, and Antonino Furnari. 2024. TI-PREGO: Chain of Thought and In-Context Learn-

- ing for Online Mistake Detection in PROcedural EGOcentric Videos. *arXiv preprint arXiv:2411.02570* (2024).
- [238] Jan Poland and Marcus Hutter. 2006. Universal learning of repeated matrix games. In *The 15th Annual Machine Learning Conference of Belgium and The Netherlands*.
- [239] Rob Powers and Yoav Shoham. 2004. New Criteria and a New Algorithm for Learning in Multi-Agent Systems. *Advances in Neural Information Processing Systems* 17 (2004).
- [240] Rob Powers and Yoav Shoham. 2005. Learning against opponents with bounded memory.. In *The Nineteenth International Joint Conference on Artificial Intelligence*. 817–822.
- [241] Arnun Pretorius, Kale ab Tessera, Andries P. Smit, Kevin Eloff, Claude Formanek, St John Grimby, Sipehelele Danisa, Lawrence Francis, Jonathan Shock, Herman Kamper, Willie Brink, Herman Engelbrecht, Alexandre Laterre, and Karim Beguir. 2021. Mava: A Research Framework for Distributed Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2107.01460* (2021). <https://arxiv.org/pdf/2107.01460.pdf>
- [242] Martin L. Puterman. 1990. Chapter 8 Markov decision processes. In *Stochastic Models*. Vol. 2. Elsevier, 331–434.
- [243] Behnam Rahdari, Peter Brusilovsky, and Branislav Kveton. 2024. Towards simulation-based evaluation of recommender systems with carousel interfaces. *ACM Transactions on Recommender Systems* (2024). <https://www.amazon.science/publications/towards-simulation-based-evaluation-of-recommender-systems-with-carousel-interfaces>

- [244] Muhammad Rahman, Jiaxun Cui, and Peter Stone. 2024. Minimum coverage sets for training robust ad hoc teamwork agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17523–17530.
- [245] Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. 2020. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems* 33 (2020), 2914–2924.
- [246] Giorgia Ramponi and Marcello Restelli. 2022. Learning in Markov Games: can we exploit a general-sum opponent?. In *The 38th Conference on Uncertainty in Artificial Intelligence*. <https://openreview.net/forum?id=S3NzSD8icx9>
- [247] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems* 33 (2020), 10199–10210.
- [248] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* 21, 178 (2020), 1–51.
- [249] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jen-

- nifer Dy and Andreas Krause (Eds.). PMLR, 4295–4304. <https://proceedings.mlr.press/v80/rashid18a.html>
- [250] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (San Francisco, California, USA) (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. <https://doi.org/10.1145/2939672.2939778>
- [251] Marc Rigter, Bruno Lacerda, and Nick Hawes. 2021. Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 13 (May 2021), 11930–11938. <https://ojs.aaai.org/index.php/AAAI/article/view/17417>
- [252] Michael J. Rosenfeld, Reuben J. Thomas, and Maja Falcon. 2018. How Couples Meet and Stay Together (HCMST). <https://data.stanford.edu/hcmst>. [Accessed 04-10-2023].
- [253] Emily Rosenzweig and Thomas Gilovich. 2012. Buyer's remorse or missed opportunity? Differential regrets for material and experiential purchases. *Journal of personality and social psychology* 102, 2 (2012), 215.
- [254] Steven I Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D Weisz. 2023. The programmer's assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*. 491–514.

- [255] G. Rummery and Mahesan Niranjan. 1994. On-Line Q-Learning Using Connectionist Systems. *Technical Report CUED/F-INFENG/TR 166* (11 1994).
- [256] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garð ar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Foerster. 2024. JaxMARL: Multi-Agent RL Environments and Algorithms in JAX. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 50925–50951. https://proceedings.neurips.cc/paper_files/paper/2024/file/5aee125f052c90e326dcf6f380df94f6-Paper-Datasets_and_Benchmarks_Track.pdf
- [257] Amit Sabne. 2020. XLA : Compiling Machine Learning for Peak Performance.
- [258] Swarnadeep Saha, Peter Hase, and Mohit Bansal. 2023. Can Language Models Teach Weaker Agents? Teacher Explanations Improve Students via Theory of Mind. *arXiv preprint arXiv:2306.09299* (2023).
- [259] Christoph Salge, Cornelius Glackin, and Daniel Polani. 2014. Empowerment—an introduction. *Guided Self-Organization: Inception* (2014), 67–114.
- [260] Ahmed Salih, Zahra Raisi-Estabragh, Ilaria Boscolo Galazzo, Petia Radeva, Steffen E

- Petersen, Gloria Menegaz, and Karim Lekadir. 2023. Commentary on explainable artificial intelligence methods: SHAP and LIME. *arXiv preprint arXiv:2305.02012* (2023).
- [261] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).
- [262] Truly Santika, Erik Meijaard, Sugeng Budiharta, Elizabeth A. Law, Ahmad Kusworo, Joseph A. Hutabarat, Tito P. Indrawan, Matthew Struebig, Sugeng Raharjo, Imanul Huda, Sulhani, Andini D. Ekaputri, Soni Trison, Madeleine Stigner, and Kerrie A. Wilson. 2017. Community forest management in Indonesia: Avoided deforestation in the context of anthropogenic and climate complexities. *Global Environmental Change* 46 (2017), 60–71. <https://doi.org/10.1016/j.gloenvcha.2017.08.002>
- [263] Rosario Scavo, Francesco Ragusa, Giovanni Maria Farinella, and Antonino Furnari. 2023. Quasi-Online Detection of Take and Release Actions from Egocentric Videos. In *International Conference on Image Analysis and Processing*. Springer, 13–24.
- [264] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.
- [265] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

- [266] Peter Schuster and Karl Sigmund. 1983. Replicator dynamics. *Journal of theoretical biology* 100, 3 (1983), 533–538.
- [267] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. 2018. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems* 1, 1 (2018), 187–210.
- [268] Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. 2024. Differentiable Task Graph Learning: Procedural Activity Representation and Online Mistake Detection from Egocentric Videos. *arXiv preprint arXiv:2406.01486* (2024).
- [269] Brennan Shacklett, Luc Guy Rosenzweig, Zhiqiang Xie, Bidipta Sarkar, Andrew Szot, Erik Wijmans, Vladlen Koltun, Dhruv Batra, and Kayvon Fatahalian. 2023. An Extensible, Data-Oriented Architecture for High-Performance, Many-World Simulation. *ACM Trans. Graph.* 42, 4 (2023).
- [270] Rohin Shah, Pedro Freire, Neel Alex, Rachel Freedman, Dmitrii Krasheninnikov, Lawrence Chan, Michael D Dennis, Pieter Abbeel, Anca Dragan, and Stuart Russell. 2021. Benefits of Assistance over Reward Learning. <https://openreview.net/forum?id=DFIoGDZeJIB>
- [271] Lloyd S. Shapley. 1953. Stochastic Games*. *Proceedings of the National Academy of Sciences* 39 (1953), 1095 – 1100.
- [272] Lloyd S Shapley et al. 1953. A value for n-person games. (1953).
- [273] Hassam Ullah Sheikh, Mina Razghandi, and Ladislau Boloni. 2019. Learning Distributed Cooperative Policies for Security Games via Deep Reinforcement Learning. In *2019 IEEE*

- 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. 489–494. <https://doi.org/10.1109/COMPSAC.2019.00075>
- [274] Mikhail Shirokikh, Ilya Shenbin, Anton Alekseev, Anna Volodkevich, Alexey Vasilev, Andrey V Savchenko, and Sergey Nikolenko. 2024. Neural Click Models for Recommender Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2553–2558.
- [275] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.
- [276] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2019. Learning Important Features Through Propagating Activation Differences. arXiv:1704.02685 [cs.CV] <https://arxiv.org/abs/1704.02685>
- [277] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
- [278] Arunesh Sinha. 2022. AI and Security: A Game Perspective. In *2022 14th International Conference on Communication Systems & NETWORKS (COMSNETS)*. 393–396. <https://doi.org/10.1109/COMSNETS53615.2022.9668430>
- [279] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. 2018. Stackelberg security games: Looking beyond a decade of success. IJCAI.
- [280] Glenn H Snyder. 1971. "Prisoner's Dilemma" and "Chicken" Models in International Politics. *International Studies Quarterly* 15, 1 (1971), 66–103.

- [281] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5887–5896. <https://proceedings.mlr.press/v97/son19a.html>
- [282] Kaushik Soni. 2020. locationtagger — pypi.org. <https://pypi.org/project/locationtagger/>. [Accessed 05-10-2023].
- [283] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24. 1504–1509.
- [284] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. 2021. Collaborating with humans without human data. *Advances in Neural Information Processing Systems* 34 (2021).
- [285] José Suárez-Varela, Paul Almasan, Miquel Ferriol-Galmés, Krzysztof Rusek, Fabien Geyer, Xiang Cheng, Xiang Shi, Shihan Xiao, Franco Scarselli, Albert Cabellos-Aparicio, et al. 2022. Graph neural networks for communication networks: Context, use cases and opportunities. *IEEE network* 37, 3 (2022), 146–153.
- [286] Ilia Sucholutsky and Thomas L. Griffiths. 2023. Alignment with human representations supports robust few-shot learning. In *Thirty-seventh Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=HYGnmSLBCf>

- [287] Ilia Sucholutsky, Lukas Muttenthaler, Adrian Weller, Andi Peng, Andreea Bobu, Been Kim, Bradley C. Love, Erin Grant, Iris Groen, Jascha Achterberg, Joshua B. Tenenbaum, Katherine M. Collins, Katherine L. Hermann, Kerem Oktar, Klaus Greff, Martin N. Hebart, Nori Jacoby, Qiuyi Zhang, Raja Marjeh, Robert Geirhos, Sherol Chen, Simon Kornblith, Sunayana Rane, Talia Konkle, Thomas P. O’Connell, Thomas Unterthiner, Andrew K. Lampinen, Klaus-Robert Müller, Mariya Toneva, and Thomas L. Griffiths. 2023. Getting aligned on representational alignment. arXiv:2310.13018 [q-bio.NC]
- [288] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf
- [289] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
- [290] Mingfei Sun, Sam Devlin, Jacob Beck, Katja Hofmann, and Shimon Whiteson. 2023. Trust region bounds for decentralized ppo under non-stationarity. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 5–13.
- [291] Peng Sun, Jiechao Xiong, Lei Han, Xinghai Sun, Shuxing Li, Jiawei Xu, Meng Fang,

- and Zhengyou Zhang. 2020. TLeague: A Framework for Competitive Self-Play based Distributed Multi-Agent Reinforcement Learning. arXiv:2011.12895 [cs.LG]
- [292] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) (*ICML'17*). JMLR.org, 3319–3328.
- [293] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).
- [294] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) (*AAMAS '18*). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085–2087.
- [295] Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.
- [296] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

- [297] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).
- [298] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786* (2025).
- [299] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. 2021. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 15032–15043.
- [300] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033. <https://doi.org/10.1109/IROS.2012.6386109>
- [301] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971 [cs.CL]*
- [302] Kristal K. Trejo, Julio B. Clempner, and Alexander S. Poznyak. 2016. Adapting strategies to dynamic environments in controllable stackelberg security games. In *2016 IEEE 55th*

- Conference on Decision and Control (CDC)*. 5484–5489. <https://doi.org/10.1109/CDC.2016.7799111>
- [303] Kristal K. Trejo, Julio B. Clempner, and Alexander S. Poznyak. 2018. Adapting attackers and defenders patrolling strategies: A reinforcement learning approach for Stackelberg security games. *J. Comput. System Sci.* 95 (2018), 35–54. <https://doi.org/10.1016/j.jcss.2017.12.004>
- [304] Johannes Treutlein, Michael Dennis, Caspar Oesterheld, and Jakob Foerster. 2021. A new formalism, method and open issues for zero-shot coordination. In *International Conference on Machine Learning*. PMLR, 10413–10423.
- [305] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Patek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems* 36 (2024).
- [306] William Uther. 2010. *Markov Decision Processes*. Springer US, Boston, MA, 642–646. https://doi.org/10.1007/978-0-387-30164-8_512
- [307] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large Language Models Still Can’t Plan (A Benchmark for LLMs on Planning and Reasoning about Change). *arXiv preprint arXiv:2206.10498* (2022).
- [308] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence*.
- [309] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N

- Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [310] Arthur F. Veinott. 1966. On the Optimality of (s, S) Inventory Policies: New Conditions and a New Proof. *SIAM J. Appl. Math.* (Sept. 1966).
- [311] Aravind Venugopal, Elizabeth Bondi, Harshavardhan Kamarthi, Keval Dholakia, Balaraman Ravindran, and Milind Tambe. 2020. Reinforcement Learning for Unified Allocation and Patrolling in Signaling Games with Uncertainty. *CoRR* abs/2012.10389 (2020). arXiv:2012.10389 <https://arxiv.org/abs/2012.10389>
- [312] Bernhard Von Stengel and Shmuel Zamir. 2004. *Leadership with commitment to mixed strategies*. Technical Report. Citeseer.
- [313] TM Vu, R Moezzi, J Cyrus, and J Hlava. 2021. Model Predictive Control for Autonomous Driving Vehicles. *Electronics* 2021, 10, 2593.
- [314] Christopher M Wade, Kemen G Austin, James Cajka, Daniel Lapidus, Kibri H Everett, Diana Galperin, Rachel Maynard, and Aaron Sobel. 2020. What is Threatening Forests in Protected Areas? A Global Assessment of Deforestation in Protected Areas, 2001–2018. *Forests* 11, 5 (2020), 539.
- [315] Ethan Waisberg, Joshua Ong, Mouayad Masalkhi, Nasif Zaman, Prithul Sarker, Andrew G Lee, and Alireza Tavakkoli. 2024. Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments. *Eye* 38, 6 (2024), 1036–1038.

- [316] Kai Wang, Aditya Mate, Bryan Wilder, Andrew Perrault, and Milind Tambe. 2019. Using Graph Convolutional Networks to Learn Interdiction Games. In *AI for Social Good workshop (AI4SG) at International Joint Conference on Artificial Intelligence (IJCAI) 2019*.
- [317] Qinghao Wang, Yijie Peng, and Yaodong Yang. 2022. Solving Inventory Management Problems through Deep Reinforcement Learning. *Journal of Systems Science and Systems Engineering* (Dec. 2022).
- [318] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2021. A Survey on Session-Based Recommender Systems. *ACM Comput. Surv.* 54, 7, Article 154 (jul 2021), 38 pages. <https://doi.org/10.1145/3465401>
- [319] Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, et al. 2023. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 20270–20281.
- [320] Xiang Wang, Shiwei Zhang, Zhiwu Qing, Yuanjie Shao, Zhengrong Zuo, Changxin Gao, and Nong Sang. 2021. Oadtr: Online action detection with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7565–7575.
- [321] Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. 2019. Deep Reinforcement Learning for Green Security Games with Real-Time Information. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 1401–1408. <https://doi.org/10.1609/aaai.v33i01.33011401>
- [322] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality

- assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
- [323] Zerui Wang and Yan Liu. 2024. Analyzing the Quality Attributes of AI Vision Models in Open Repositories Under Adversarial Attacks. *arXiv preprint arXiv:2401.12261* (2024).
- [324] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.
- [325] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).
- [326] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.
- [327] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. 2003. Inequalities for the L1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep* (2003), 125.
- [328] Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makovychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. 2022. EnvPool: A Highly Parallel Reinforcement Learning Environment Execution Engine. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 22409–22421. https://proceedings.neurips.cc/paper_files/paper/

2022/file/8caaf08e49ddb6694fae067442ee21-Paper-Datasets_
and_Benchmarks.pdf

- [329] J. Wijngaard and J.C. Wortmann. 1985. MRP and inventories. *European Journal of Operational Research* 20, 3 (1985), 281–293.
- [330] Guoquan Wu, Miguel Ángel de Carvalho Servia, and Max Mowbray. 2023. Distributional reinforcement learning for inventory management in multi-echelon supply chains. *Digital Chemical Engineering* (March 2023).
- [331] Yue Wu, Xuan Tang, Tom M Mitchell, and Yuanzhi Li. 2023. Smartplay: A benchmark for llms as intelligent agents. *arXiv preprint arXiv:2310.01557* (2023).
- [332] Jing Xu, Fangwei Zhong, and Yizhou Wang. 2020. Learning Multi-Agent Coordination for Enhancing Target Coverage in Directional Sensor Networks. *Advances in Neural Information Processing Systems* 33 (2020).
- [333] Lily Xu, Elizabeth Bondi, Fei Fang, Andrew Perrault, Kai Wang, and Milind Tambe. 2020. Dual-Mandate Patrols: Multi-Armed Bandits for Green Security. *arXiv preprint arXiv:2009.06560* (2020).
- [334] Lily Xu, Andrew Perrault, Fei Fang, Haipeng Chen, and Milind Tambe. 2021. Robust reinforcement learning under minimax regret for green security. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 161)*, Cassio de Campos and Marloes H. Maathuis (Eds.). PMLR, 257–267. <https://proceedings.mlr.press/v161/xu21a.html>

- [335] Xianliang Yang, Zhihao Liu, Wei Jiang, Chuheng Zhang, Li Zhao, Lei Song, and Jiang Bian. 2023. A Versatile Multi-Agent Reinforcement Learning Benchmark for Inventory Management. arXiv:2306.07542 [cs.AI] <https://arxiv.org/abs/2306.07542>
- [336] Yaodong Yang and Jun Wang. 2021. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. arXiv:2011.00583 [cs.MA]
- [337] Shunyu Yao. 2024. *Language Agents: From Next-Token Prediction to Digital Automation*. Ph. D. Dissertation. Princeton University.
- [338] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems* 35 (2022), 20744–20757.
- [339] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. 2019. On the (in) fidelity and sensitivity of explanations. *Advances in neural information processing systems* 32 (2019).
- [340] Kenny Young and Tian Tian. 2019. MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments. *arXiv preprint arXiv:1903.03176* (2019).
- [341] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. arXiv:2103.01955 [cs.LG]
- [342] Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. 2018. Deep Reinforcement

- Learning for Green Security Game with Online Information. <https://www.aaai.org/ocs/index.php/WS/AAAIW18/paper/view/17362/15586>
- [343] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.
- [344] Yuanqing Yu, Chongming Gao, Jiawei Chen, Heng Tang, Yuefeng Sun, Qian Chen, Weizhi Ma, and Min Zhang. 2024. EasyRL4Rec: A User-Friendly Code Library for Reinforcement Learning Based Recommender Systems. *arXiv preprint arXiv:2402.15164* (2024).
- [345] Jaleh Zand, Jack Parker-Holder, and Stephen J. Roberts. 2022. On-the-fly Strategy Adaptation for ad-hoc Agent Coordination. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (Virtual Event, New Zealand) (AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1771–1773.
- [346] Daniel J Zarin, Nancy L Harris, Alessandro Baccini, Dmitry Aksenov, Matthew C Hansen, Claudia Azevedo-Ramos, Tasso Azevedo, Belinda A Margono, Ane C Alencar, Chris Gabris, et al. 2016. Can carbon emissions from tropical deforestation drop by 50% in 5 years? *Global change biology* 22, 4 (2016), 1336–1347.
- [347] Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. 2024. ProAgent: Building Proactive Cooperative Agents with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*

38, 16 (Mar. 2024), 17591–17599. <https://doi.org/10.1609/aaai.v38i16.29710>

- [348] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM on Web Conference 2024*. 3679–3689.
- [349] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5579–5588.
- [350] Qizhen Zhang, Chris Lu, Animesh Garg, and Jakob Foerster. 2022. Centralized Model and Exploration Policy for Multi-Agent RL. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1500–1508.
- [351] Weijia Zhang, Hao Liu, Fan Wang, Tong Xu, Haoran Xin, Dejing Dou, and Hui Xiong. 2021. Intelligent Electric Vehicle Charging Recommendation Based on Multi-Agent Reinforcement Learning. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM. <https://doi.org/10.1145/3442381.3449934>
- [352] Xiaohan Zhang, Yan Ding, Saeid Amiri, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2023. Grounding classical task planners via vision-language models. *arXiv preprint arXiv:2304.08587* (2023).
- [353] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New

- Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.
<https://doi.org/10.1561/15000000066>
- [354] Stephen Zhao, Chris Lu, Roger B Grosse, and Jakob Foerster. 2022. Proximal Learning With Opponent-Learning Awareness. *Advances in Neural Information Processing Systems* 35 (2022), 26324–26336.
- [355] Wentao Zhao, Jiaming Chen, Ziyu Meng, Donghui Mao, Ran Song, and Wei Zhang. 2024. Vlmpe: Vision-language model predictive control for robotic manipulation. *arXiv preprint arXiv:2407.09829* (2024).
- [356] Wei Zhao, Pengxiang Ding, Min Zhang, Zhefei Gong, Shuanghao Bai, Han Zhao, and Donglin Wang. 2025. VLAS: Vision-Language-Action Model With Speech Instructions For Customized Robot Manipulation. *arXiv preprint arXiv:2502.13508* (2025).
- [357] Yu-Sheng Zheng and Awi Federgruen. 1991. Finding optimal (s, S) policies is about as simple as evaluating a single policy. *Operations research* 39, 4 (1991), 654–665.
- [358] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinu Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. LIMA: Less is more for alignment. *arXiv preprint arXiv:2305.11206* (2023).
- [359] Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Yong Yu, Jun Wang, and Weinan Zhang. 2023. MALib: A Parallel Framework for Population-based Multi-agent Reinforcement Learning. *Journal of Machine Learning Research* 24, 150 (2023), 1–12. <http://jmlr.org/papers/v24/22-0169.html>

- [360] Yilun Zhu, Joel Ruben Antony Moniz, Shruti Bhargava, Jiarui Lu, Dhivya Piraviperumal, Yuan Zhang, Hong Yu, and Bo-Hsiang Tseng. 2024. Can Large Language Models Understand Context? *arXiv preprint arXiv:2402.00858* (2024).