A HYBRID MACHINE LEARNING AND AGENT-BASED MODELING APPROACH TO EXAMINE DECISION-MAKING HEURISTICS

By

Paul Cummings
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computational Social Science

Committee:

______________________________ Dr. Hamdi Kavak, Committee Chair

______________________________ Dr. Andrew Crooks, Committee Member

______________________________ Dr. William Kennedy, Committee Member

______________________________ Dr. Michael Eagle, Committee Member

______________________________ Dr. Jason Kinser, Department Chair

______________________________ Dr. Donna M. Fox, Associate Dean,
Office of Student Affairs & Special
Programs, College of Science

______________________________ Dr. Fernando Miralles-Wilhelm, Dean,
College of Science

Date:__________________________ Fall Semester 2020
George Mason University
Fairfax, VA

A Hybrid Machine Learning and Agent-Based Modeling Approach to Examine Decision-Making Heuristics

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Paul Cummings
Master of Arts in Interdisciplinary Studies
George Mason University, 2018
Bachelor of Arts
Boston University, 1992

Director: Hamdi Kavak, Professor
Department of Computational and Data Sciences

Fall Semester 2020
George Mason University
Fairfax, VA

DEDICATION

This is dedicated to my loving family Naomi, Kaeya, Mia, and Hero Dogs Captain.

He's such a good boy.

ACKNOWLEDGEMENTS

I would like to thank my wife, Naomi, my children Kaeya and Mia, and my loving parents Anne and Jim Cummings, who always guided my educational success. 'I'd like to thank my advisor Dr. Hamdi Kavak and Dr. Andrew Crooks that stood by me in my concept, as well as my other esteemed committee members Dr. Bill Kennedy and Dr. Michael Eagle. Finally, 'I'd like to thank those research scientists who came before me, as I stand on the brilliant work of those before me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

Advantage Actor-Critic	A2C
Artificial Neural Networks	ANN
Asynchronous Advantage Actor-Critic	A3C
Artificial Intelligence	AI
Agent-Based Model	ABM
Computer Generated Forces	CGF
Cross-Validation	CV
Deep Learning Important FeaTures	DeepLIFT
Deep Q-Network	DQN
Deep Reinforcement Learning	DRL
Finite State Machine	FSM
Learning-based Actor-Interpreter State Representation	LAISR
Local Interpretable Model-agnostic Explanation	LIME
Long Short-Term Memory	LSTM
Machine Learning	ML
Multi-Agent Informational Learning Processes	MAILP
Multi-Agent Reinforcement Learning	MARL
Multi-Agent Systems	MAS
Markov Decision Process	MDP
Partial Dependency Plot	PDP
Partially Observable Markov Decision Process	POMDP
Policy Proximal Optimization	PPO
Recurrent Neural Network	RNN

Shapley Additive exPlanations	SHAP
Explainable Artificial Intelligence	XAI

DEFINITIONS

Given many of the terms and definitions related to agent modeling and machine learning may have different connotations, this section begins with a set of definitions that articulate what is meant by a particular term.

Agent Based Model (ABM): A computational model for representing and studying a social system consisting of autonomous, interacting, goal-oriented, bounded-rational set of actors that use a given rule set and are situated in an environment. An ABM consists of three main components: agents, rules, and environments where agents are situated (Cioffi-Revilla, 2014).

Reinforcement Learning (RL): The term reinforcement learning describes a dynamically learning, trial and error method to maximize the outcome, while deep reinforcement learning (DRL) is learning from existing knowledge and applying it to a new data set (Williams, 1987).

Bounded Rationality: A concept that describes decision-making and planning under resource limitations, which is an adaptation strategy for coping with our innate lack of perfect rationality. In other words, social institutions are causally explained by Bounded Rationality (Simon, 1957b).

Bounded Optimality: The concept refers to algorithms that maximize utility where the optimal solution is constrained by its technology architecture, computing power, and the task environment (Horvitz, 1988).

Emergent Behavior: Emergent behavior is a type of global behavior that arises from many agents interacting in a system, but is not attributable to any particular agent (Privosnik, 2002)

Heuristic: A heuristic is a cognitive strategy that is composed of building blocks, typically three: search rules that specify where to look for information, stopping rules that specify when to end search, and decision rules that specify how to make a final decision.

Fast-and-frugal Heuristics: A simple, task-specific decision strategies that are part of a decision maker's repertoire of cognitive strategies for solving judgment and decision tasks (Gigerenzer and Todd, 1999).

Heuristic vs. Strategy: The term heuristic and strategy can be used somewhat interchangeably, where heuristics are precisely defined as *strategies derived from previous experiences with similar problems*.

Satisfice: To be satisfied with a minimum or merely satisfactory level of performance, profitability, etc., rather than a maximum or optimum level (Simon, 1956).

LAISR Model: Term is used when referencing both Actor (deep reinforcement learning) and Interpreter (AI interpretation methods). See Chapter 3.

DRL-Agent: Term referring only to the deep reinforcement learning agent

ABSTRACT

A HYBRID MACHINE LEARNING AND AGENT-BASED MODELING APPROACH TO EXAMINE DECISION-MAKING HEURISTICS

PAUL CUMMINGS, PhD

George Mason University, 2020

Dissertation Director: Hamdi Kavak

Agent-Based Models (ABMs) have become more widespread over the last two decades allowing researchers to explore complex systems composed of heterogeneous entities. Although ABMs have proven effective for generating simple rules over homogenous and heterogeneous agent types to observe emergent behaviors, several challenges exist. One, typical ABMs are limited in the representation of cognition and learning to maximize their actions based on current (and future) rewards of being in a particular state. Two, ABMs are not designed to produce their own behaviors that can be interpreted by the designer. Although agents may act upon code generated by the model designer, their local and global responses are not easily interpretable. Additionally, ABMs do not decompose behaviors into information rooted in cognitive processing, specifically satisficing or fast-and-frugal heuristics. To address these challenges, this dissertation presents a model and methodology called the Learning-based Actor-Interpreter State Representation (LAISR), where agents use Deep Reinforcement Learning (DRL) to

generate strategies to maximize for current and future states. Due to the agent's behavior representation as a deep neural network (DNN), explainable artificial interpretation (XAI) methods are used to decompose DNN features into simple but satisficing strategies (heuristics). The results of this work demonstrate an approach that bridges machine learning with that of the social sciences where agents can build their own optimal and boundedly rational strategies. This methodology is demonstrated across several homogeneous and heterogeneous agent-based models. The implications of this work demonstrate significant steps towards how machine learning-enhanced ABMs can be used to develop novel and optimal decision strategies, enhance human behavior modeling, and provide a bridge between social science and artificial intelligence research.

1 INTRODUCTION

Agent-Based Models (ABMs) have grown in their application over the last two decades partly because such a style of the model can represent complex systems with minimal knowledge of parameter values or without fully knowing the optimal parameter states to describe the real-world environment (Li et al., 2013). Not only do ABMs allow us to study complex systems, but they also provide an intuitive and realistic description of the behavior of such systems. Agent-based modeling and simulation platforms offer architectures of varying complexity for the agents, where reactive agents are very simplistic, reacting to environmental stimuli often without any long-term reasoning; finite-state machines require the scripting of all of the possible states of the agents and the corresponding behaviors; cognitive agents offer a more flexible description of behaviors in terms of goals and plans (Kennedy, 2012). Agent-based modeling has proven itself invaluable to the social science community, including work in conflict (Epstein, 2006; Bhavnani and Miodownik, 2009), segregation (Schelling, 1971; Bruch and Mare, 2006), evolutionary biology (Holland, 1992; Nowak, 2006), and ecology (DeAngelis and Mooij, 2005; Heckbert et al., 2010). Often cognitive theories are difficult to address from an agent-based modeling perspective; it is challenging to ensure they are calibrated, generated, and accurately evaluated in reference to a real-world system (Cioffi-Revilla, 2014).

1.1 Challenges with ABMs

There are three main challenges which this dissertation addresses in reference to modeling rationality in ABMs.

1.1.1 Challenge 1: ABMs are limited in their ability to find optimal strategies

By their design, ABM models are limited in their abilities as information processors and are boundedly rational (Simon, 1957b). ABMs act in accordance with a set of simple variables and parameters given by the modeler and make decisions that are primarily scripted or coded. Notably, they are not able to learn strategies on their own. Ideally, agents learn to maximize their ability to select actions based on current (and future) rewards of being in a particular state.

1.1.2 Challenge 2: ABMs are not easily interpreted

ABMs are not designed to produce their own behaviors that can be interpreted by the designer. Although agents may take action based on code generated by the model designer, their behaviors are often difficult to analyze. Additionally, given agents act within complex systems, often producing emergent properties, the process of interpreting agent behaviors can be even more challenging.

1.1.3 Challenge 3: ABMs strategies are not decomposable to simple heuristics

Given the social science community's interest in modeling decision processes, ideally, we would like to be able to provide techniques that mimic human cognitive processes, i.e., quick decisions, particularly when working with complex data. Although these ‘heuristics’ made may not necessarily be optimal, they can aid our understanding of how humans acquire and employ decision strategies.

1.2 An Example ABM to Illustrate the Challenges: The Bat-Frog Predation Model

In this example, two distinct agent types, the bat (predator) and frog (prey), must interpret and respond to their environment in order to survive. Here are the basic rules:

- The frog would like to mate but must be careful not to be eaten by a bat.
- The frog can only mate if it calls out to a mate, so it must be careful not to give itself up.
- The bat, on the other hand, can hear mating calls but only at specific frequencies and when it is awake (primarily at night).
- The bat must be careful not to eat *all* the frogs; otherwise, there is no mating for its future meals.

In pseudo programming, one might write frog and bat programming that would do the following:

Frog Model: The frog selects a list of frequencies to sing to and selects a set of times it speaks to mating frogs. If it sees a bat, stop singing and choose an escape behavior. If male, sing to the mate, and if female, listen for mating call, go to mate.

```
def frog ():
        #select low and high frequencies
        var song frequency = list {low frequency, high frequency}


        #select a start and end time
        var time activity = list {start time, end time}


        #update function
    def Update ()
    #take actions if seen other predators or prays


        If {bat-seen}
            [stop-singing, hide, jump in the river, move quickly left and right, etc].
            If {mate-seen}
            [if male->sing to mate; if female (and hear singing) -> go to mate]
```

Bat Model: The bat selects a list of frequencies to listen to and selects a set of times it listens for mating frogs. If it hears a frog, fly to it and eat it. Randomly choose not to eat the frog in order to ensure they can procreate.

```
    def bat ():
        var hear song frequency = list {low frequency, high frequency}
        var time activity = list {start time, end time}
        #update function
    def Update ()
        If (hear song frequency == frog frequency)
        {fly to frog; eat frog}
        Random (1.0) < .2 {leave frogs to procreate}
```

In this pseudo-code example, the Bat-Frog model designer may generate a set of parameters that could be tested through a parameter sweep and examine how each predator and prey may react in the environment.

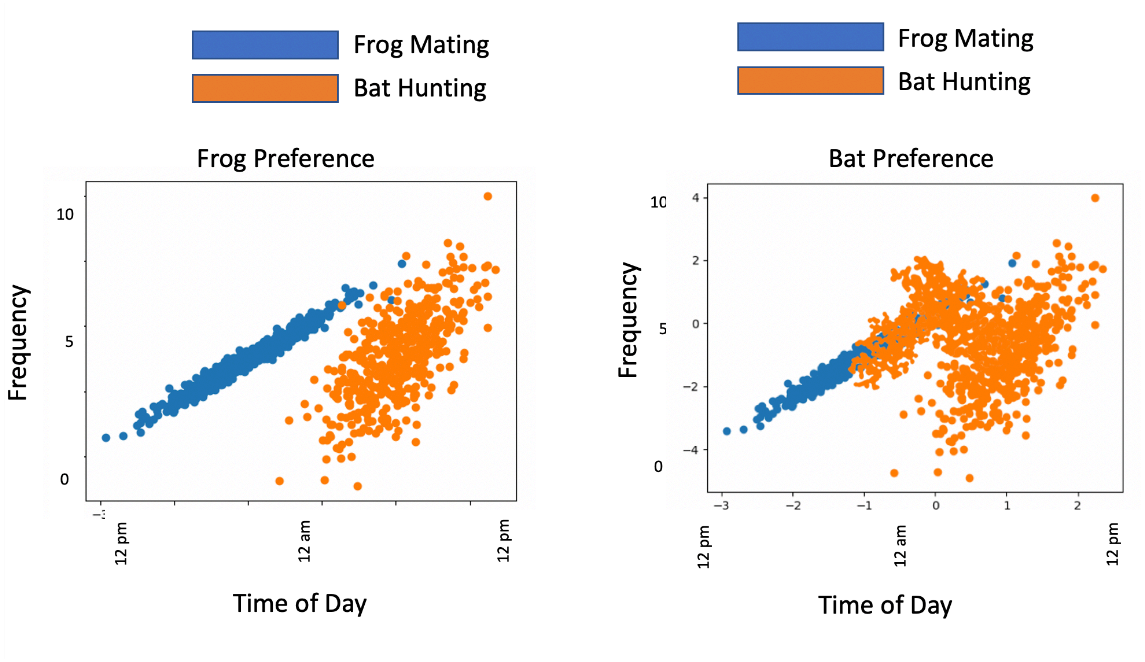


Figure 1: Frog Preferences (a) and Bat Preferences (b)

After running the model several times, Figure 1 (a) illustrates that the frog may have a preference to never overlap with bat hunting in order to maximize its survival. On the other hand, in Figure 1 (b), the bat would choose to overlap sometimes with the frog in order to eat but would still leave enough frogs for future meals. From here, the observer might deduce the following strategy:

- Bat: eat but don't eat everything
- Bat: eat when frogs are out
- Frog: stay hidden unless you need to eat

- Frog: procreate

1.2.1 Complex Bat-Frog Model

The complexity of the problem shall now be enhanced. Let's assume that frogs mate if the female frog can hear the male frog sing at a certain frequency where the sonic frequency of calls allows mate recognition and generally matches the female tympanic range (Ryan, 1988). Additionally, the bat can hear certain frequencies, and the frog must continually update its song frequencies in order to not get spotted by the bat.

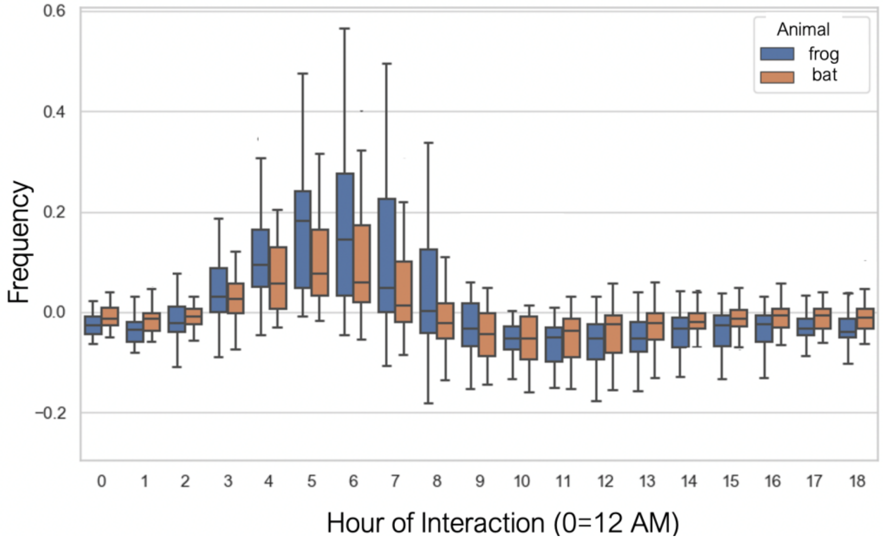


Figure 2: Bat Behavior Relative to Frog

In Figure 2, the bat must learn to adapt its hearing to recognize frog frequencies at certain times of the day. The frog, in turn, must modulate its frequencies so it may survive and mate. As one can imagine, the set of conditions and variables can get particularly challenging to determine as the number of parameters increases. In the model, the conditions are at state s of the environment at time t; the bat must determine the action (song frequency f and hour of interaction$t$) of the frog. The frog may select to continuously update its frequency modulation every few days to keep the bat guessing its whereabouts, and with this, the bat-frog model gets even more complex. In the simple bat-frog example, agents are bestowed behaviors coded in advance by the modeler (Sen, 1999). But for a more dynamic set of conditions as in the complex example, behaviors must be learned such that actions are taken to achieve maximum rewards for making a decision in a particular state. How does an agent learn such a process? Let us refer back to the challenges discussed in section 1.1.

Imagine that instead of generating a set of specific actions, we can endow each agent with reward signals that the agents must strive for to achieve their respective goals. If the agent takes an action A at state S and is given a reward R for being in that state, then over time, we can develop a set of optimal action-state pairs which lead to the highest reward (Sutton and Barto, 1998).

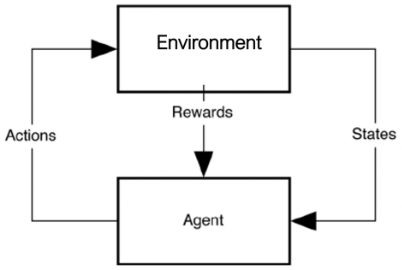


Figure 3: Agent Takes Actions in a State and is Rewarded

This *learning* strategy imbues agents with a type of continuous updating and refining its approach to optimizing actions within its environment. The continuous self-governing learning process also removes a considerable amount of coding for the model developer (i.e., it is up to the agent to discover competent behavior), and eventually, the agent learns to select appropriate actions based on the ability to maximize its rewards in a particular state.

In the example, imagine that now that the agent has generated its behaviors through utility maximization, we can deconstruct its behaviors (preferably through explainable artificial intelligence methods) into sets of actions taken given specific conditions. These features can be generated into a set of strategies (heuristics) used to describe the behavior of the agent.

1.3 Research Questions

RQ 1: What methods can aid in the design, development, and analysis of hybrid ABM and reinforcement learning system in efforts to address challenges in ABM modeling?

This research question addresses the importance of how agents could learn independently to build their own learned strategies using deep reinforcement learning (DRL) techniques. In classical ABM design, agents are built with discrete rules and then self-organize into systems that perform tasks and potentially generate emergent properties (Hutchinson and Gigerenzer, 2005). DRL models, on the other hand, are not designed with discrete rules; these rules must rather be learned through deep reinforcement learning mathematical principles. But, certainly, we do not want to throw the baby out with the bathwater. ABMs have some very important characteristics, such as allowing us to generate simple rules over homogenous and/or heterogeneous agent types and observe emergent behavioral changes. ABMs are also fairly easy to decipher at their fundamental level. It is therefore important to consider a research question that includes the values of both traditional ABMs and features of DRL that can aid in the enhancement of the proposed research question.

RQ 2: What AI-based research techniques can help to deconstruct behaviors of the proposed agent model into decision strategies?

If RQ 1's goal is to use deep reinforcement learning methods to build optimal policies for each agent, DRL models must also be deconstructed. Unfortunately, DRL agents use artificial neural networks (ANN) to build their internal representation of the environment, which are challenging to decipher due to the "black box" nature of these structures (Kamruzzaman et al., 2010)). Explainable AI (XAI) techniques provide some support in our understanding of the ANN. However, a more sophisticated process needs to be developed to further decompose XAI data in order to uncover specific strategies generated by the DRL agent. This process will help us develop a more precise equivalent to the Hutchinson et al. (2005) and Todd & Gigerenzer (1999) cognitive heuristic.

Based on these research questions, this dissertation presents a novel approach to designing and developing a formal ABM and machine learning hybrid model called the *Learning-based Actor-Interpreter State Representation* (LAISR). The model is tested within the bounds of computational social sciences theory to include Schelling's model, homogeneous and heterogeneous model design, and ultimately an analytical method to deconstruct its behaviors into strategies. The design, development, and analysis of the approach are discussed over the course of the following chapters.

1.4 Methodology and Dissertation Organization

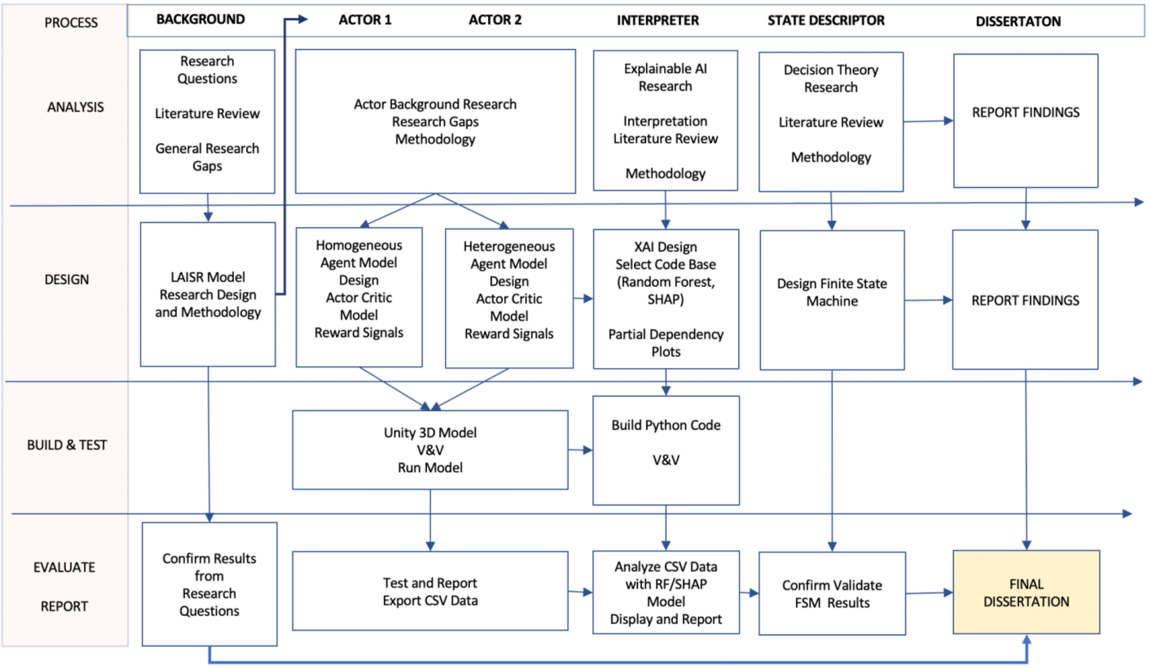


Figure 4: Methodology

The following methodology provides an overarching view of the procedures used in this dissertation. Across the top, from left to right, there are six development sections in this dissertation: An initial set of research questions are gathered based on gaps related to social ABM design challenges. From here, a research design plan was generated to include the design of the LAISR model. For each portion of the LAISR model (Homogeneous Actor 1, Homogeneous Actor 2, Interpreter, State Descriptor), an Analysis, Design, Build & Test, Evaluate, and Report process was followed. Steps were written into respective sections of the dissertation, as were the results of the experiments. For each section having a software component to it, a verification and validation section was developed to ensure the code was properly implemented and validated. Through each successive "analysis to evaluation" period, data were collected and included within a report, which was added to this dissertation. All code was uploaded to GitHub and is available for review in order to replicate the entirety of this process.

1.4.1 Dissertation Chapters

Background introduces the background to this work, which includes research in a) Rationality, b) Utility, c) Interpretation, and d) Heuristics, and gaps leading to the design of this model. Additional background chapters in Artificial Intelligence and Explainable Artificial Intelligence provide detail about theoretical, mathematical, and computational methods for developing the LAISR model.

LAISR Experiments: Homogeneous Models builds on the previous chapter by developing two agent-based models with multiple agent types that must compete for resources within an environment. This chapter highlights how LAISR agents can generate an appropriate strategy in relation to their goals (see section 3.2.3).

LAISR Experiments: HeterogeneousModel attempts to generate *a set* of heterogeneous multi-agent reinforcement learning agents. The principal concept is to evaluate how the LAISR agent explores its own learning strategy when in competition with an alternate agent type.

LAISR Experiments: Advanced Explainable Artificial Intelligence investigates the complexity of machine learning interpretation, including a research section on new methods of AI interpretability. This section also contains a section on the development of *a State Descriptor* (see 4.10).

Finally, *Discussion and Future Work* provides a look into the LAISR model from the social science perspective, present areas of future research, and discusses disciplines that can benefit from this work in the future. Finally, the new field of Inverse Generative Social Science (IGSS) is introduced, which in some form could provide additional insights into new generative methods at play in social system theory.

2 BACKGROUND

We want the [reinforcement learning] agent to explore to find changes in the environment. As in the earlier exploration/exploitation conflict, there probably is no solution that is both perfect and practical, but simple heuristics are often effective. - Sutton and Barto (1998)

2.1 ABMs in the Social Science Community

ABMs have long been a method for examining collective and emergent properties of complex systems. These properties emerge from local behaviors of a multitude of agents in social science contexts (Bonabeau, 2002; Miller et al., 2007). In order to identify emergent behavior in an ABM, it is first necessary to identify local rules that generate the *intended behavior* at system a larger scale (Fehérvári, 2010). ABMs can simulate the evolutionary attributes of complex systems environments with a large number of parameters for many time steps (Li et al., 2013). ABMs have some very important characteristics, such as allowing us to generate simple rules over homogenous and/or heterogeneous agent types and observe emergent behavioral changes. ABMs are also fairly easy to decipher at their fundamental level; this feature becomes more difficult as ABMs aggregate their behavior, but the building blocks can be fundamentally simple.

But ABMs have their deficiencies as described in section 1.1 *Challenges with ABMs*. In order to overcome these shortcomings, I will outline four areas of research that are necessary to begin the enhancement of the ABM model: a) *Rationality*, b) *Utility*, c) *Interpretation*, and d) *Heuristics*. Each of these areas is addressed from two areas of research: Decision-making theory and Artificial Intelligence Research.

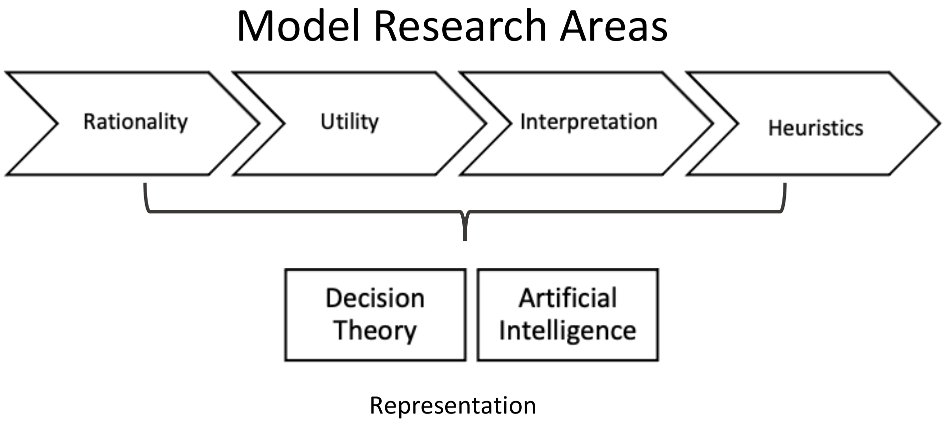


Figure 5: Four Model Research Areas and Decision Theory and AI Representation

2.2 Rationality

A considerable amount of work has gone into the concept of rationality decision-making (Malpas, 2012). Humans make decisions under a variety of conditions and make these decisions with information and resource constraints. Researchers in social science theory has tried to define decision-making from the perspective of rationality. Currently, there are two primary decision-making approaches: Bounded Rationality (Allard, 2003) and Rational Decision-Making. A rational thinker is predicted to reach their highest results in the selection of decisions made while considering values, attributes, and risk preferences.

2.2.1 Bounded Rationality Decision-Making

Bounded rationality (Simon, 1957b) assumes that information for both human cognition and AI agents is imperfect, and we accommodate this constraint within the theory of bounded rationality.

Table 1: Bounded Rationality Representations

	Decision Theory	Artificial Intelligence
Definition	Due to the mind's limitations, humans must approximate methods to handle most tasks (Simon, 1990).	Modeling behavior requires reward functions to maximize expected future rewards with no previous knowledge of the environment (Sutton and Barto, 1998)
Methods and Models	Mental processing, environmental structure/constraints Satisficing Friedman model	Kahneman Cognition Architecture Russell and Norvig Bounded and Globally Rational Model

2.2.2 Representing Bounded Rationality

Decision theory and AI are endowed with representations of bounded rationality. Where Decision Theory sees it as a cognitive processing limitation, AI makes the assumption that an agent does not have full awareness of its environment. Over time it must learn to generate its own set of actions that it believes will create maximum utility. In Table 1, bounded rationality is defined by Simon (1990) and represented as one of several cognitive representations (satisficing, fast-and-frugal). The AI equivalent is algorithms such as the Bellman Equation that maximize future rewards with minimal knowledge about the environment.

2.2.2.1 Bounded Rationality: Decision Theory

Herbert Simon (1957b) was one of the first to argue that human beings are bounded in their ability to be completely rational. Simon suggested that humans behave in an irrational manner due to a lack of important information that would help them define the problem before making a decision. Simon envisioned bounded rationality as two interlocking components: the limitations of the human mind and the structure of the environments in which the mind operates.

The first component of his vision means that models of human judgment and decision-making should be built on the mind's ability to process rather than on all known rationality. In many real-world situations, optimal strategies are unknown or unknowable (Simon, 1987). Even in a game such as chess, where an optimal (best) move does, in fact, exist at every point, no strategy can calculate that move in a reasonable amount of time (either by human minds or computers), despite the well-defined nature of the possibilities to be searched. In less well-defined natural situations, the hope of identifying a useable optimal strategy is even further diminished. Because of the mind's limitations, humans must approximate methods to handle most tasks (Simon, 1990). These methods include recognition processes that largely obviate the need for further information search, heuristics that guide search and determine when it should end, and simple decision rules that make use of the information found.

The second component of Simon's view of bounded rationality, environmental structure, explains *when* and *why* simple heuristics perform well. Broadly stated, "the task is to replace the global rationality of economic man with the kind of rational behavior that is compatible with the access to information and the computational capacities that are actually possessed by organisms, including man, in the kinds of environments in which such organisms exist." (Simon 1955a: 99). Others made this point before Simon (e.g., Brunswik, 1943) several times during his life. Balancing the quality of a decision against its costs was popular in economics (Stigler, 1961). To this day, it remains common to formulate boundedly rational decision-making as a constrained optimization problem. Milton Friedman's *as if* methodology (Friedman, 1953) similarly uses models that ignore contributing factors supporting decision-making. Todd & Gigerenzer (2007) describe how we make inferences about the world around us with limited information and processing power. Whereas several models assume rationality assumes a type of "all-knowingness" (Ibid). These models are in conflict with what we understand about both rationality and processing power.

2.2.2.2 *<u>Satisficing</u>*

Satisficing (Simon, 1957a) is a boundedly rational decision-making strategy whose goal is determining a satisfactory or acceptable result, rather than an optimal solution. Instead of putting maximum exertion toward attaining the ideal outcome, satisficing focuses on pragmatic effort when confronted with tasks. This is because aiming for the optimal solution may necessitate a needless expenditure of time, energy, and resources. For example, it can be used to choose between two competing available objects, where options are limited, rather than searching across a large space of choices (Gigerenzer, 2000).

2.2.3 Bounded Rationality: AI

AI-Based Bounded Rationality is essentially the AI analog to human bounded rationality. The links between Bounded Rationality and machines are often seen in AI research on two-player zero-sum games with perfect information such as Checkers, Chess, and Go. Within AlphaGo and AlphaGo Zero, the algorithms are designed where "thinking" is bounded rational in that it is not designed to undertake a full search of the decision tree (Lee, 2019). The super-human gameplay demonstrated by AlphaGo and AlphaGo Zero also demonstrates that *computational rationality* can lead to super-human Artificial Intelligence(AI) even though it is not able to observe all possible states (Lee, 2019) bounded rationality and AI do not often appear together in research articles. Economists discussing bounded rationality rarely mention AI and vice versa. Lee (2019) noted this is surprising given Herbert Simon was a pioneer in both bounded rationality and AI. Simon undertook research on both bounded rationality and AI simultaneously in the 1950s, and these interests persisted throughout his research career.

Simon's research in the 1950s was critical to the growth of computational science, social science theory, and AI. The pursuit of modeling or replicating human behavior has developed two general camps which can be applied to the social sciences: Artificial Intelligence and Cognitive Science. Cognitive science is designed to replicate aspects of human behavior and emotion, where AI replicates human behavior and, with some effort, may surpass human intelligence (Ibid).

2.2.3.1 *Boundedly and Globally Rational AI Models*

Russell and Norvig's (2010) represented AI behaviors as a matrix of Thinking-Acting and Bounded-Global categories. In the first category, AI resolved to develop computational models that create 'systems that think' which are similar to that of humans. These mechanisms include language, knowledge representation (and memory), reasoning, and learning (Lee, 2019). In the second category, machines are required to act like humans. They do not need to possess a human-like mechanism. In the third category, the focus is on machines that think rationally in terms of mathematical logic. The fourth and last group relates to machines that take actions that are optimal (rational) but may not be based on logical reasoning. This dissertation's interest resides in the first column (see Table 2).

Table 2: AI and Bounded Rationality (Source: Adapted from Russell and Norvig (2010), Figure 1.1, p.2.)

	Bounded Rationality	**Global Rationality**
Thinking (Mental Process)	1.Thinking Humanly Limitations in learning, memory and computation (learning, self-learning)	3. Thinking Rationally (Super intelligent?) (Universal Turing Machine)
Acting (Action)	2.Acting Humanly Not globally optimal outcome/action (Brute-force search)	4. Acting Rationally Globally Optimal outcome/action (Non-Halting UTM) (Incomputable)

2.2.3.2 The Kahneman Cognition Architecture

Determining a satisfactory result has been observed by Kahneman (2003) and others as part of a spectrum of methods of intuition (System 1) and reasoning (System 2) where "intuitive thoughts seem to come spontaneously to mind, without conscious search or computation, and reasoning is a slower less intuitive but more process-driven response, the reasoning is done deliberately and effortfully" (Kahneman, 2003). Their cognitive architecture is illustrated in Figure 6. Kahneman also states that these two systems do not exist independently. For example, System 2, through the process of repetition, can eventually move into an unconscious fast state. Kahneman goes on to discuss how deep learning is much closer to System 1, where it finds patterns to assemble behaviors, yet there is no causality or meaning attributes in deep learning and until that is solved.

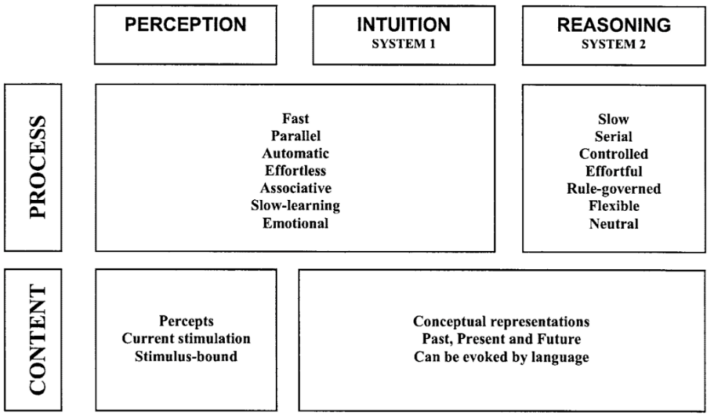


Figure 6: Kahneman Architecture of Cognition: Source (Kahneman, 2003)

2.2.3.3 Gaps in Bounded Rationality Research

Heuristics-and-biases tradition has been criticized by Gigerenzer et al. (2005) and others for being too focused on how heuristics lead to errors. The critics argue that heuristics can be seen as rational in an underlying sense. According to this perspective, heuristics are good enough for most purposes without being too demanding on the brain's resources. Another theoretical perspective sees heuristics as fully rational in that they are rapid, can be made without full information, and can be as accurate as more complicated procedures. Heuristics are useful in a variety of circumstances but can also be cognitively biased (Korteling et al., 2018; Simon, 1955; Broadbent, 1958; Kahneman, 1973, 2003; Norman & Bobrow, 1975). Although the proposed criticisms are well-founded, there is very little research on how to automatically generate heuristics. Rather, most of the work has been on how modelers build heuristics into their agent models (by hand).

2.2.3.4 Perfect Rationality Decision-making

Coming from economic research, the theory of rational decision-making aims to connect varied phenomena into a single body of mathematical social theory. Perfect rationality, coming from rational choice theory, orders the decisions on the basis of subjective expected utility (von Neuman, 1953). Here, *homo economicus* (Mill, 1848) is perfectly rational and has complete knowledge, while his economic choices, guided by rationality, are self-contained in the economic sphere without affecting other aspects of the individual, such as the emotions or being influenced by the environment.

2.3 Utility

Table 3: Expected Utility Definitions and Measurement

	Decision Theory	Artificial Intelligence
Definition	A weighted average of the utilities of each of its possible outcomes,	Given a set of states, select actions that maximize current and future rewards
Methods and Models	The utility of an outcome measures the extent to which that outcome is preferred, or preferable, to the alternatives.	Action, state, outcome representations Maximal set of possibilities, P, of which each state, act, or outcome is a subset.

This section discusses Utility and Expected Utility and how they are aligned from the perspective of Decision Theory and Artificial Intelligence (AI). Utility in Decision Theory provides the cognitive foundations for the model, where AI provides a means to implement the theory in an AI mathematical model.

2.3.1 Maximizing Utility: Decision Theory

The Stanford publication Normative Theories of Rational Choice: Expected Utility (2014) introduces the expected utility hypothesis. Bernoulli (1738) is a method for an agent to optimize its current and future states. The expected utility of an action is "a weighted average of the utilities of each of its possible outcomes, where the utility of an outcome measures the extent to which that outcome is preferred, or preferable, to the alternatives" (Briggs, 2019). The expected utility has some general characteristics where *outcomes are* preferred over future states; these *states* are often outside the decision maker's control.

2.3.2 Maximizing Utility: AI

The expected utility acts as a bridge between bounded rationality, heuristics, and machine learning methods. Within the expected utility, there are three essentials: actions, states, and outcomes.

- States, actions, and outcomes are all sets of possibilities. There is a maximal set of possibilities,P, of which each state, act, or outcome is a subset.
- The set of actions, the set of states, and the set of outcomes are all sub-components of P, i.e., actions and states are individualized so that every possibility in P is one where an agent attempts to maximize its expected reward and chooses an action based on current state conditions. The expected utility of an action A depends on two qualities:
    a) The value of each outcome is measured by a *utility*.
    b) The probability of each outcome conditional on A.

In exactly one state, the agent performs exactly one action, and exactly one outcome ensues.[1]Navarro-Martinez et al. (2018) designed a "choice under risk" satisficing model, which puts Expected Utility Theory (EUT) in a boundedly rational framework. The decision-maker gathers evidence for and against options favoring one option satisfies the desired level of confidence.

[1] The author also assumes for the moment that, given a state of the world, each act has exactly one possible outcome.

2.4 Interpretation

An aspect of the model that deals with explaining the model decisions, so they are represented as heuristics, is interpretation. Decision theory interpretation modeling is often cognition and declaration-focused on where activities are described or contextualized by an operator. On the other hand, AI interpretation is highly algorithm-based; in reference to deep learning, AI interpretation is usually the interpretation of the 'black box' neural network substructure.

Table 4: Interpretation Definitions and Measurement

	Decision Theory	Artificial Intelligence
Definition	Dedicated to explaining and interpreting decisions and modeling of the decision maker's preferences	Dedicated to developing procedures to make ML understandable to the user
Methods and Models	Decision Support Systems (DSS) Multiple Criteria Decision-making (MCDM)	IF-THEN rules Recurrent Neural Network (RNN) Sequence Prediction DeepLIFT LIME/SHAP

2.4.1 Interpretation: Decision Theory

Interpretable Decision Theory can be thought of as solving a problem by a) explaining your reasoning to yourself and/or someone else by b) starting from a high abstraction level, and c) breaking the problem into smaller sub-problems. Although the literature on interpretable Decision Theory is sparse, the Analytic Hierarchy Process

(AHP) (Saaty, 1999) was devised as a weighted sum, where the primary tasks are broken into sub hierarchical tasks. The weights are gathered from specialists using a pair-wise operator that generates a matrix; this is then changed into a set of weights by a normalized Eigenvector.

2.4.2 Interpretation: AI

This section is covered in detail in section *3.4: Background in Explainable Artificial Intelligence (XAI)*, where the dissertation presents IF-THEN rules (3.4.1), Recurrent Neural Network (RNN) (3.4.2), Sequence Prediction (3.4.3), DeepLIFT (3.4.4), and LIME/SHAP (3.4.5.1) algorithms.

2.5 Heuristics

The intention now is to address how Decision Theory heuristics (fast-and-frugal) can be adopted into AI techniques such as fast-and-frugal trees (FFTs) and finite state machines (FSM) (see sections 2.5.2.1- 2.5.2.2).

Table 5: Heuristics Definitions and Measurement

	Decision Theory	Artificial Intelligence
Definition	Simple strategies or mental processes that are used to quickly form a judgment make decisions and find solutions to complex problems.	Set of discrete instructions that are used to solve a problem within an action space
Methods and Models	Processes Fast-and-frugal	Discrete decision process that selects between one or more solutions based on simple pre-determined probability.

	Take the best Take the first Rule-of-thumb	Fast-and Frugal Decision Trees Finite State Machines

2.5.1 Heuristics: Decision Theory

The heuristic rule-of-thumb strategy minimizes the time it may take to make decisions without stopping to consider the next course of action (Todd & Gigerenzer, 1999). Heuristics are essentially fast, efficient processes, i.e., rules of thumb. A heuristic is composed of building blocks, typically three: search rules that specify where to look for information, stopping rules that specify when to end search, and decision rules that specify how to make a final decision. (Hutchinson et al., 2005; Todd & Gigerenzer, 1999) propose that the brain does not work in intricate probabilities and functions; actually, the mind is bounded by its own strategies and works using *fast-and-frugal* heuristics. Essentially, human decision-making processes can be demonstrated using a minimal number of heuristics without full knowledge of time and information. These models provide crude computational abilities and do not participate in the domain of probability (Gigerenzer and Todd, 1999).

2.5.1.1 <u>Fast-and-Frugal Heuristics</u>

Fast-and-frugal heuristics (Gigerenzer and Todd, 1999) are simple to execute, limited information, search, and computation rules generated by humans. They are building blocks that demonstrate searching (search rule), termination of searching stopped (stopping rule) and a processing step towards a decision (decision rule).

Determining these heuristics is an important part of the work that is being presented. The fact that behavior can be shaped through the use of rewards provided by my model is a core tenet of both behavioristic psychologies as well as reinforcement machine learning.

Fast-and-frugal heuristics generate adaptive choices by minimizing the use of time, knowledge, and computational processing. Numerous studies have investigated the extent to which models of fast-and-frugal heuristics accurately describe people's choices and decisions and the underlying cognitive processes(Hertwig, Hoffrage, & Martignon, 1999). Examples include resource allocation (Hertwig, Davis, & Sulloway, 2002), classification (Berretty, Todd, & Martignon, 1999), preferential selection (Brandstätter et al., 2006, 2008), and decision-making (Dhami, 2003).

2.5.2 Heuristics: AI

Forster (1999) argued fast-and-frugal heuristics couldn't emerge from an underlying complexity in a process that is driven by machine intelligence and AI. Yet, it remains unclear how people's decision processes compare to resource-rational behavior. Leider et al. (2017) modeled the decision method as an arrangement of computations to generate an optimal decision process.

2.5.2.1 Fast-and-Frugal Trees (FFT)

FFTs are a type of decision tree with consecutively ordered lines, where every line has two branches where one is an exit value (Martignon et al., 2003). The final line has two exit points stating a decision is always made. FFTs are fast; like cognitive heuristics, decision-making usually occurs within a very few lines. Green & Mehr (1997) and Martignon et al. (2008) have demonstrated that predictive precision of FFTs is comparable to machine learning-based decision trees. As the main proponent of FFTs, Luan (2011) believes FFTs may be more cognitively realistic as compared to complicated machine-learning algorithms given the speed the brain processes.

2.5.2.2 Finite State Machine (FSM)

Another type of state representation is the Finite State Machine (FSM). One can consider the finite state machine as a triple $\boldsymbol{M = (S, R, t)}$, where $\boldsymbol{S}$ is a finite set of states., $\boldsymbol{R}$ is a finite set of symbols called the alphabet., $\boldsymbol{t: S \times A \rightarrow S}$ is the transition function. The inputs to this function are the current state and the last input symbol. While the function value $\boldsymbol{v(s, x)}$ is the state, the automaton goes from state $\boldsymbol{s}$ after reading symbol $\boldsymbol{x}$. Then the resultant FSM should mimic behaviors of its ML counterpart. FSMs may be more accurate when describing predictive methods of behavior that may involve more complex nodes than the FFT model (see section 2.5.2.1), although results may not be as indicative of cognitive heuristics given they are not as 'fast' (Luan, 2011). From

Figure 7, one can see that the structure of representation may also help to determine which would be a dominant choice. For example, when the conditional states are binary, and choices can be decomposed into a simple tree structure, the FFT may be a better selection, whereas, if each state has multiple conditional probabilities, it may be more prudent to choose the FSM approach.

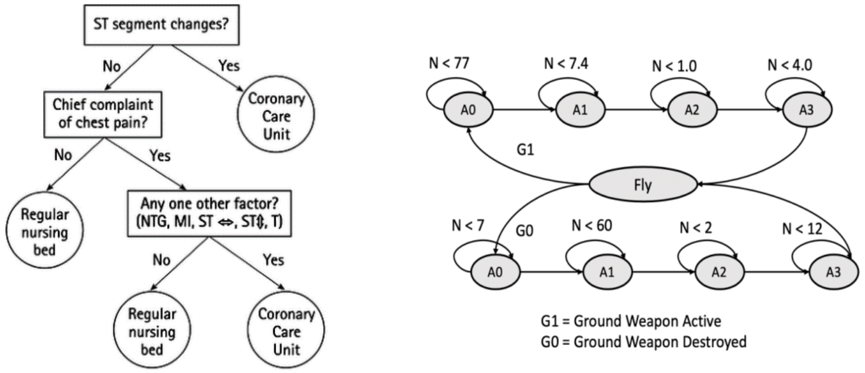


Figure 7: Fast-and-frugal Tree (Source: Gigerenzer & Gaissmaier, 2011) and Finite State Representation

2.6 Summary

The background outlined the primary components of the model, a) Rationality, b) Utility, c) Interpretation, and d) Heuristics, noting that given there are two disciplines (Decision Theory, Artificial Intelligence) that must be integrated in order to build the model. The next two sections describe the background in machine learning and AI-explainability research necessary to build the agent model.

3 BUILDING BLOCKS OF THE DRL-AGENTAND EXPLAINABLE ARTIFICIAL INTELLIGENCE

Definition: DRL-Agent

The DRL-Agent is an ABM that employs Deep Reinforcement Learning to generate maximum value for being in a particular state. The following chapter will discuss mathematical and computational underpinnings necessary to understand and develop the DRL-Agent.

3.1 Artificial Intelligence

AI is often defined as a computer system with the ability to perform tasks commonly associated with intelligent beings. As this definition somewhat problematically requires us to define intelligence and is inconveniently repetitive, Artificial Intelligence is now commonly defined as a scientific discipline, as the activity that creates machines that can function appropriately and with foresight in their environment. The first explicit definition of

AI was suggested in a funding proposal to the Rockefeller Foundation in 1955. It was based on the "conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it." (Tuomi, 2018) This early definition rapidly led to deep controversies. In practice, the early developers of AI interpreted intelligence and thinking as mechanical processing of logical statements, thus, in effect, defining human intelligence as computation of truth values.

3.1.1 Machine Learning

Machine Learning is a subset of the term Artificial Intelligence, provides automated methods that can detect and learn patterns in data and use them to achieve some tasks (Christopher, 2006; Murphy, 2012). Three types of machine learning tasks are explained:

- *Supervised learning* is the task of inferring a classification or regression from labeled training data.
- *Unsupervised learning* is the task of drawing inferences from datasets consisting of input data without labeled responses.
- *Reinforcement learning* (RL) is the task of learning how agents ought to take sequences of actions in an environment in order to maximize cumulative rewards.

3.1.2 Supervised Learning Methods

Although not directly applicable to the LAISR model, supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. (Russell and Norvig, 2010) It infers a function from labeled training data consisting of a set of training examples (Mohri et al., 2012). In supervised learning, each example is a pair consisting of an input object and the desired output value (or supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario allows for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to alternate data sources. Although supervised learning is a field of dynamic research, the proposed dissertation focuses on the second area of research, unsupervised learning. Supervised learning is based on training data that has been labeled, usually by humans, so that the network weights can be adjusted when the labels for training data are wrongly predicted. After a sufficient number of examples are provided, the error can, in most cases, be reduced to a level where the predictions of the network become useful for practical purposes. For example, if an image detection program tries to differentiate between cats and dogs, during the training process, someone needs to tell the system whether a picture contains a cat or a dog.

3.1.3 Unsupervised Learning

Unsupervised learning is a branch of machine learning that learns from test data that has not been labeled, classified, or categorized (Hinton and Sejnowski, 1999). Instead of responding to feedback, unsupervised learning identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. Alternatives include supervised learning and reinforcement learning. A central application of unsupervised learning is in the field of density estimation in statistics (Smola et al., 2008) though unsupervised learning encompasses many other domains involving summarizing and explaining data features. It could be contrasted with supervised learning by saying that whereas supervised learning intends to infer a conditional probability distribution, unsupervised learning intends to infer an a priori probability distribution.

3.1.4 *Reinforcement Learning (RL)*

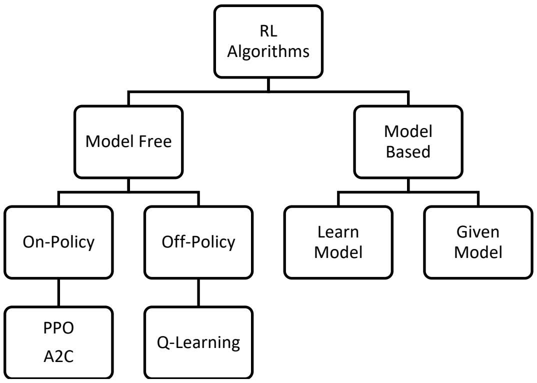


Figure 8: Reinforcement Learning Taxonomy

RL exists between supervised and unsupervised learning. In traditional supervised learning, there is a target label for each training example, and in unsupervised learning, there are no labels, where in reinforcement learning, there are sparse and time-delayed rewards (Pathak, 2017)—based only on those rewards the agent has to learn to generate optimal behaviors within the environment(Sutton et al., 1995). Reinforcement learning can be understood using the concepts of agents, environments, states, actions, and rewards, all of which are explained in the equations below. Capital letters tend to denote sets of things, and lower-case letters denote a specific instance of that thing, e.g., A is all possible actions, while a is a specific action contained in the set.

- **Action (A):** A is the set of all possible moves the agent can make.
- **Discount factor (γ)**: The discount factorγ is multiplied by future rewards as discovered by the agent in order to dampen these reward's effects on the agent’s choice of action.

- **Environment (€)**: The€world through which the agent moves. The environment takes the agent's current state and action as input and returns as output the agent’s reward and its next state
- **State (S):** State s is a concrete and immediate situation in which the agent finds itself, i.e., a specific place and moment, an instantaneous configuration that puts the agent in relation to other significant elements of the environment.
- **Reward (R):** A reward $r$is the feedback by which measures the success or failure of an agent’s actions.
- **Policy (π):** The policy π is the strategy that the agent employs to determine the next action based on the current state. It maps states to actions, the actions that promise the highest reward.
- **Value (V):** The value v is the expected long-term return with discount, as opposed to the short-term reward R. Vπ(s) is defined as the expected long-term return of the current state under policy π. I discount rewards or lower their estimated value the further into the future they occur.
- **Q-value or action-value (Q):** Q-value is similar to value, except that it takes an extra parameter, the current action a. Qπ(s, a) refers to the long-term return of the current state s, taking action a under policy π. Q maps state-action pairs to rewards.

In RL, an agent interacts with an environment and uses the experience to optimize a decision-making policy. In a standard RL formulation, the agent aims to max-following the policy after first using action to advance. With these definitions in hand, we can briefly review the deep RL algorithms. RL enables agents to learn policies for task performance based on rewards received over a sequence of trials (Sutton et al., 1995). A reinforcement learning (RL) agent learns by interacting with its dynamic environment

(Kaelbling, 1998). At each time step, the agent perceives the state of the environment and takes an action, which causes the environment to transit into a new state. A scalar reward signal evaluates the quality of each transition, and the agent has to maximize the cumulative reward along the course of interaction.

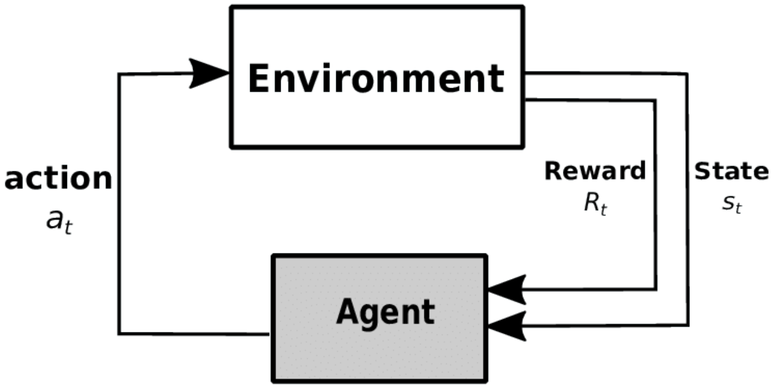


Figure 9: Reinforcement Learning Cycle (Source: Recreated from Sutton and Barto, 1998)

3.1.4.1 <u>The Markov Decision Processes (MDP)</u>

A primary component within the model is the MDP. MDPs are often used to model sequential decision processes in machine learning systems. This *policy* maximizes the accumulated expected reward is then considered optimal and can be learned from sampling. Unfortunately, model parameters are often assessed from noisy data (Mannor et al., 2007; Roy et al., 2017). This second type of uncertainty can often degrade the performance of the optimal strategy and thereby affect the model's prediction. Neto (2005) discussed the concept of the MDP and discussed the concept eloquently in this introduction to agent reinforcement learning where it is defined as a tuple (S, A, T, R) where:

- A is an action set.
- S is a state space.
- $T: S \times A \times S \rightarrow [0, 1]$ is a transition function defined as a probability distribution over the states. Hence, we have $T(s, a, s_0) = Pr\{s_{t+1} = s_0 \mid s_t = s, a_t = a\}$. s_{t+1}represents the state of the process at time t+1, st the state at time t, and at the action taken after observing state st.
- $R: S \times A \times S \rightarrow R$ is a reward function representing the expected value of the next reward, given the current state s and action and the next state a_0: $R(s, a, s_0) = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s_0\}$. In this context r_{t+1}represents the immediate payoff of the environment to the agent at time $t + 1$ (Bellman, 1957; Howard, 1960; Bertsekas, 1995; Sutton and Barto, 1998; Puterman, 1994). The MDP acts on the environment with action a, in state s, and waits for the response of the environment, in the form of the following state s_0 and a real number representing the immediate reward the agent receives by choosing to perform a in s. The task of deciding which action to choose in each state is done by a policy function.

3.2 The Policy

Generally, a policy is a collection of probability distributions, one for each trace of the process $\pi(s_t, a_t - 1, s_{t-1}, a_{t-2}, \dots) \in PD(A)$ defining the probability that each action is chosen for that particular trace of the system. However, there is no need to consider other than Markovian policies because the MDP itself is Markovian by construction – it is sufficient to define the policy for each state of the MDP.

3.2.1.1 Partially Observable Markov Decision Processes (POMDP)

The POMDP is a dominant operation of RL is the recurrent interaction between an agent and a Markov Decision Process. Specifically, the interaction assumes that the agent knows everything about the current state of world: there is no notion of hidden information (aside from not knowing the causal rules or reward structure). The POMDP does not make this assumption. In either case, the agent interacts indefinitely with its world, trying to update its beliefs about what exists in the world and how to take actions to maximize reward. A POMDP is a Markov model that attaches unobservable states to observations. The agent can perform actions that maximize their reward. However, the agent cannot directly observe the system state, but at each discrete point in time, the agent makes observations that depend on the state. The agent uses these observations to form a belief of what state the system currently is. This belief is called a belief state and is expressed as a probability distribution over the states. The solution of the POMDP is a policy prescribing which action is optimal for each belief state. The POMDP framework is broad enough to model a variety of real-world sequential decision-making problems. A discrete-time POMDP can be described as a 7-tuple $(S , A , T , R , \Omega, O , \lambda)$, where

- $S = \{s_1, s_2, \ldots, s_n\}$ is a set of states,
- $A = \{a_1, a_2, \ldots, a_n\}$ is a set of actions,
- T is a set of conditional transition probabilities $T\ (s_0 |\ s, a)$ for the state transition $s \rightarrow\ s_0$.
- $R : S \times A \rightarrow R$ is the reward function,
- $\Omega = \{o_1, o_2, \ldots, o_n\}$ is a set of observations,

- O is a set of conditional observation probabilities $O(o \mid s_0, a)$, and
- $\lambda \in [0, 1]$ is the discount factor.

At each time period, the environment is in some state $s \in S$. The agent chooses an action $a \in A$, which causes the environment to transition to a state $s_0 \in S$ with probability $T\ (s_0 \mid s, a)$. At the same time, the agent receives an observation $o \in \Omega$ which depends on the new state of the environment with probability $O(o \mid s_n, a)$. Finally, the agent receives a reward $R(s, a)$. Then the process repeats. The goal is for the agent to choose actions at each time step that maximizes its expected future discounted reward. Within the RL model, agents evaluate their respective states and generate an action based on maximizing rewards. Given the state of the environment, the agent needs to pick the best action to maximize rewards. Through reinforcement learning's trial and error, it accumulates knowledge through these (state, action) pairs, as in, it can tell if there would be a positive or negative reward given a (state, action) pair. This value is referred to as the Q-value, which a value of being in a particular state, and choosing a particular action, otherwise stated as $Q\ (state, action)$. An elementary way to store this knowledge would be a Q-value table populated with $states$, $actions$, and $Q\ (state, action)$ pairs. Once enough data is collected, the trained model can generally determine the best Q-state to select.

3.2.2 Model-Free vs. Model-Based Reinforcement Learning

There are two primary areas of RL, Model-Free and Model-Based. Model-based RL uses an experience value to build a model of the transitions and outcomes in the environment. Appropriate actions are then chosen by searching or planning in this world model. Model-free RL uses experiences to learn $(state/\ action\ values\ or\ policies)$, which can potentially achieve equivalent optimal behaviors but without estimation or use of a world model. Given a policy, a state has a value, defined in terms of the future utility that is expected to accrue starting from that state.

3.2.3 Model-Free Methods: Off-Policy and On-Policy Methods of RL

The two most popular classes of model-free reinforcement learning algorithms are *Off-Policy* and *On-Policy Methods*. They are described in sections 3.2.4 and 3.2.5, with examples from each that are relevant to this work.

3.2.4 Off-Policy Methods

Off-policy reinforcement learning learns about one policy, π_1, while the reward observations are generated by the action sequence of another policy, π_2.

3.2.4.1 Q-Learning

Q-learning (an off-policy method) is a type of value iteration method that aims at approximating the Q function, while policy gradient (an on-policy method) is a method to optimize in the action space directly. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. It does not require a model of the environment and can handle problems with stochastic transitions and rewards without requiring adaptations. For any finite Markov decision process (FMDP), Q-learning finds a policy that is optimal in the sense that it maximizes the expected value of the total reward overall successive steps, starting from the current state (Melo, 2007). Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly random policy (Ibid). "Q" names the function that returns the reward used to provide the reinforcement and can be said to stand for the "quality" of an action taken in a given state (Matiisen, 2015). Q-Learning is designed to explore rather than respond to a fixed set of rules designed by the model developer.

Q-learning estimates a state-action value function (Q_SA) for a target policy π that deterministically selects the action of the highest value. Instead of directly parameterizing a policy, Q-value learning methods estimate the Q-function as $\mathrm{Q}(\mathrm{s}, \mathrm{a}; \theta)$. The greed y policy selects the (discrete)action maximizing value:$\mathrm{a}^* =$ $\mathrm{argmaxaQ}(\mathrm{s}, \mathrm{a}; \theta)$. Exploration can be performed using a greedy policy, which chooses a uniform random action with probability and otherwise uses the greedy action. By itself-policy nature, Q-learning permits repeated training use of samples. It can determine information from the environment and receive rewards for performing those actions (Neto et al., 2005). Within the model, the agents are not told which actions to take but instead must discover which actions generate the highest rewards by trying out experimenting with "actions within" in the environment.

3.2.5 On-Policy-Based Methods

Policy-based methods seek to optimize the policy space. In policy gradient methods, the policy is directly parameterized. Although ABMs have been studied in the form $\pi(\mathrm{a}|\mathrm{s}; \theta)$,where π is a probability distribution over actions a when observing states, as parameterized by θ, a neural network. The agent exercises the policy in the environment, recording experiences. Periodically, it uses the samples to update θ by estimating the gradient $\nabla\theta E[Rt]$. Typically, the agent then discards these samples and repeats, optimizing the policy iteratively.

3.2.5.1 Advantage Actor-Critic (A2C)

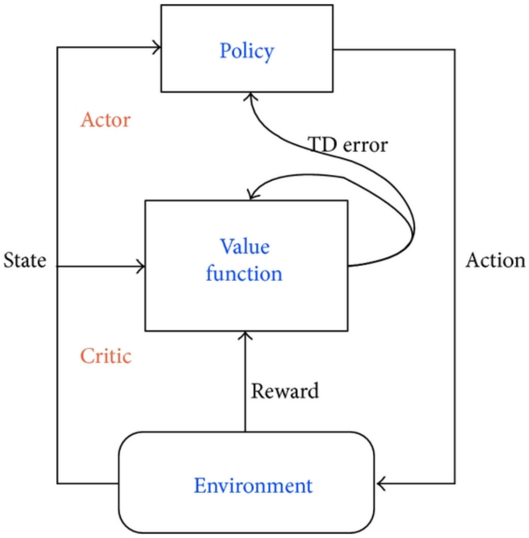


Figure 10: Actor-Critic (Source: Recreated from Sutton and Barto, 1998)

In advantage of actor-critic, the policy gradient is computed as $E[\nabla\theta\log\pi(a_t|s_t;\theta)(Rt - V(s_t))]$.The agent estimates $V(s_t)$ from the data, for instance, using separate output conjunction with ML, very little work has been done to address how hybrid models can use aspects from the same network used for π.$(Rt - Vt)$estimates the advantage: $A(s,a) = Q(s,a) - V(s)$.Rt is computed using the discounted sum of as many future returns as are observed in a given batch, up to rtmax, both ABMS and is bootstrapped with $V(stmax + 1)$, appropriately discounted. The estimator $V(s;\theta)$ is trained using, e.g., a squared-error loss simultaneously to π. In A3C (Mnihetal., 2016), a separate actor-learner threads sample environment steps and update a centralized copy of the parameter asynchronously to each other. In (batched) A2C, which performs similarly to A3C (see, e.g. (Schulman et al., 2017)), separate environment instances are sampled, but the learner is single-threaded. It gathers all data into one mini-batch to compute the gradient.

3.2.5.2 Proximal Policy Optimization (PPO)

The PPO introduced by Schulman et al. (2017) is a policy gradient technique for reinforcement learning, which does the following: the PPO samples data through actions and interactions within the simulated environment. Here it must optimize a specific (objective) function using stochastic gradient ascent. Where policy gradient approaches perform a single update per sample; thus, the objective function creates a set of mini-batch updates. These updates are more generalized, and have improved sample complexity (Juliani, 2018).

3.3 Deep Reinforcement Learning (DRL)

RL models discussed in this chapter have some substantial challenges to overcome. Most notably, it is possible for decisions to become too complex for the traditional reinforcement learning method. When the simulation becomes complicated, the knowledge space can become intractable, and it no longer becomes feasible to store all (state, action) pairs in a table. In intuitive terms, even a small difference in states is still a distinct state. In lieu of storing and looking up every distinct state, RL can employ a neural network that predicts the reward for an input (state, action). Alternatively, neural networks can predict value and policy-based methods.

3.4 Background in Explainable Artificial Intelligence (XAI)

XAI is dedicated to developing procedures to make deep learning more understandable and thus build an understanding of the model to the user (Park and Hendricks, 2018; Zintgraf, Cohen, Adel, & Welling, 2017). While the accuracy obtained by neural networks may be more precise than human experts, the complexity of the neural network structure makes it very difficult, if not impossible, to uncover complex attributes of its network connection. This term is deemed a “black box” problem (Kamruzzaman et al., 2010). Torrey et al. (2005) attempted to build rules to describe an RL policy by using *source-task* models, otherwise described as a “decomposition strategy” (Andrews, 1995), i.e., a method where model mechanics affect rules that are extracted. Craven (1996) defines rule extraction as follows: "given a trained neural network and the data on which it was trained, a description of the network that closely approximates the network's predictive behavior." A Rule extraction approach can also help to validate a neural network (Ibid).

3.4.1 IF-THEN rules:

The general form of the IF-THEN rule is designed to state simply if a state can be applied to a condition and stated as true, then the state can be added to a particular class. The IF-THEN developer can decide the level of resolution necessary to build the IF-

THEN decision tree. Andrews et al. (1995) generated an IF-THEN neural network taxonomy algorithm. The Decomposition procedure works by dividing the network into neurons. Each result is then combined to represent the entire network. DIFACON-miner (Özbakır, Baykasoğlu, and Kulluk (2010)) was designed to generate IF-THEN rules from an artificial neural network. The rule creation process places each repetition in a non-sequential process. The uniqueness of the decompositional model is that rule generation is performed at the same time as neural network training. Using evolutionary algorithms, Dorado et al. (2002) created a "black-box" that used Genetic Programming (GP) model to develop a rule-extraction approach for artificial neural networks, irrespective of network structure.

3.4.2 Time-Based Behavior Evaluation: Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) are key methods to model time-based data. Hidasi et al. (2016) have contributed substantial work in the development of recommender systems, although new and very relevant work has been done in the use of explainable AI using RNNs (Tan et al., 2016). A subset of RNN, the long short-term memory (LSTM) model is a type of artificial recurrent neural network (RNN) used to evaluate temporal based information (Lee, 2019). Wang et al. (2016) developed a novel semantic perception model that suggests an LSTM-based series prediction. The approach improves the prediction performance by uncovering semantics hidden in the observed sequences.

3.4.3 Sequence Prediction

Sequence prediction models evaluate models from the perspective of arrangement. Y. Liu et al. (2016) suggested a time-based approach to integrating evolving preferences with interval assessment. When considering spatial and temporal contexts sequentially, Liu et al. also extended RNN and proposed a method that could model local temporal and spatial contexts in each layer. Q. Liu et al. (2017) suggested an approach that evaluated behavioral sequences using transition matrices.

3.4.4 DeepLIFT

DeepLIFT (Deep Learning Important FeaTures) is a recursive calculation method for supervised deep learning (Shrikumar, 2017). DeepLIFT decomposes the prediction value of a neural network by backpropagating the out of each neuron in the network to every feature of the input network. Shrikumar (2017) notes that typical perturbation-based approaches (e.g., LIME/SHAP) and certain gradient-based approaches fail to model saturation, where his backpropagation process is designed to handle such characteristics. Similar layer-wise relevance propagation models have been developed to interpret the predictions of deep networks (Bach et al., 2015).

3.4.5 Marginal Contribution Algorithms:

3.4.5.1 LIME and SHAP

LIME (Ribiero, 2016) and SHAP (Lundberg et al., 2017) approximate how features affect prediction by perturbing instances of data sets and b) analyzing these perturbations on a classification system (i.e., black box output). Due to their generalization, they explain several classifications; i.e., neural networks in medical, law, and behavioral science (Elshawi et al., 2019; Whitmore et al., 2019; Ibrahim et al., 2019). SHAP and LIME are both standard approaches to model explainability. SHAP (Shapley Additive exPlanations) uses Shapley values defined as the “average marginal contribution of a feature value over all possible associations.” (Shapley, 1953); due to the exhaustive search of SHAP, it can ensure consistent accuracy across variables. In fact, the SHAP model is used for interpretation within this effort.

SHAP is an additive feature attribution model where explanations are stated as a set of linear features. SHAP computes a Shapley value based on a general game-theoretic model (Lundberg et al., 2016). Let us say, for example, there is a basketball game with three players on each team, Mark, Jill, and Bob. Mark alone scores 60 points in the game. Jill, a seasoned player, gets 80 points. When Bob plays together, they each score 90 (totaling 180). Would it be fair to say that Bob is a key factor for the team’s success? This is not entirely accurate; rather, it is best to consider how different combinations of the three players can contribute to the team’s success. As Lundberg et al. (2019) discuss the model, he defines three important characteristics of the SHAP model.

- Local accuracy: feature attributions sum should be equivalent to the explainable output of the model.
- Missingness: features that are not explained or are missing to not cause a change in the model.
- Consistency: altering a model where a variable has a greater influence will never reduce the importance of that feature.

3.4.5.2 Partial Dependency Plots

The PDP shows the marginal effect one or more features have on the predicted outcome of a machine learning model. The plot can also show whether the relationship between the target and a feature is linear, monotonic, or more complex. PDPs may be an appropriate tool for generating averaged features with a target (Friedman, 2001).

4 DEVELOPING THE LEARNING-BASED ACTOR-INTERPRETER STATE REPRESENTATION MODEL

4.1 Introduction

This chapter introduces a new agent model and methodology called the *Learning-based Actor Interpreter model* (LAISR) (Cummings et al., 2020). *Learning-based* refers to the fact that the agent attempts to maximize actions taken in the scenario based on what it perceives to be the best set of actions or policy. Within LAISR, there is the *Actor*, *Interpreter*, and *State Representation*. The *Actor*, in reference to the first research question (see section1.3) the uses DRL (see section 3.3) to derive its behaviors using a simple reward system. In reference to research question two, the *Interpreter* decomposes behaviors into a model that can be expressed in terms of its behavioral strategies (see section 1.3). Finally, the *State Representation* generates a readable or referenceable output of the Interpreter model.

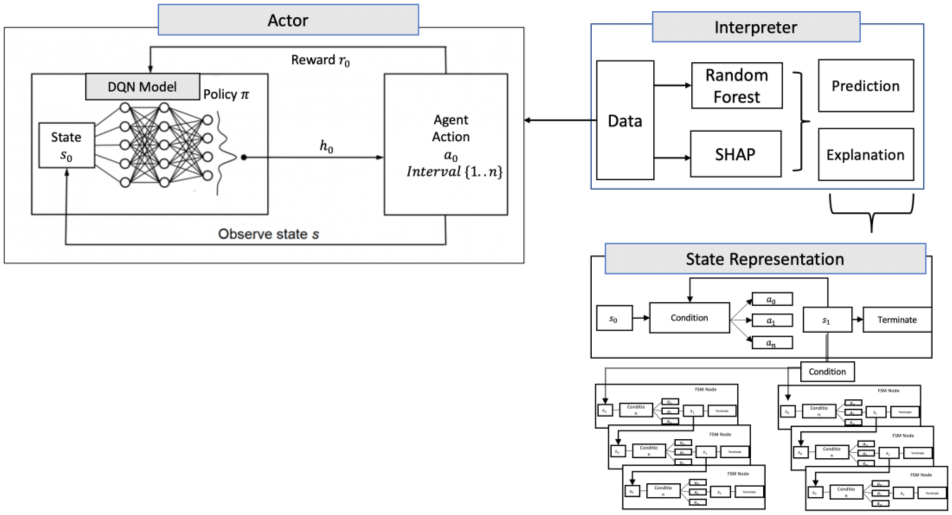


Figure 11: LAISR Model (Source: Cummings and Crooks, 2020)

4.2 The Actor – A Deep Reinforcement Learning Agent

The *Actor* portion of the LAISR model uses a type of neural network enhanced reinforcement learning (RL) called Deep Reinforcement Learning (DRL) (see section 3.3). DRL-based agents build strategies (or policies) that lead to the highest long-term expected rewards (Sutton and Barto, 1995). Much like human decision-making, DRL agents construct and learn their own knowledge with minimal input from the model designer. Using DRL, the Actor generates its own optimal policy based on expected future rewards; these behaviors are then *explained* from the perspective of the Interpreter.

4.3 The Interpreter

While efficient and versatile, the DRL's (see section 3.3) use of neural networks to approximate its policy is generally unintelligible to the viewer (Lee, 2019), thereby reducing its value as a research behavior evaluation tool. These models are essentially black boxes (Castelvecchi, 2016), i.e., the nature of neural networks makes it nearly impossible to inspect how the algorithm is accomplishing its function. Even for a network with only a single layer, it is quite challenging to understand how patterns arise due to the complexity of the network (Kamruzzaman et al., 2010). The mechanisms that solve DRL models are hidden within an interconnected network of input, hidden, and output layers. The *interpreter* attempts to evaluate the state of the Actor model and then predicts its behavior in the form of a set of states and conditions.

4.4 State Representation: Developing Strategies

The State Representation provides a method to translate the Interpreter's interpretation into a human or code readable format. This representation is designed to generate a fully usable and coherent representation of the data returned from the Interpreter. This may come in the form of an excel spreadsheet, a python implementation, or a more abstract form such as a finite state machine representation or a form of declarative knowledge. This final state representation also provides a means to decompose interpretations into a result that defines the characteristics of a strategy. For example, the Behavior State representation in the Bat-Frog example (see section 0) might be a representation of a set of states and the probability of being in a particular state. It may also be something more concrete such as a code representation, including variables and methods that describe the bat and frog behavior. Nevertheless, the approach provides some flexibility as to how detailed the designer may wish the model to be. Here, the developer can select the level of parsimony and near decomposability (Simon, 1957a) necessary to accurately model and present behavior results.

4.5 Developing, Running, and Analyzing a LAISR Model

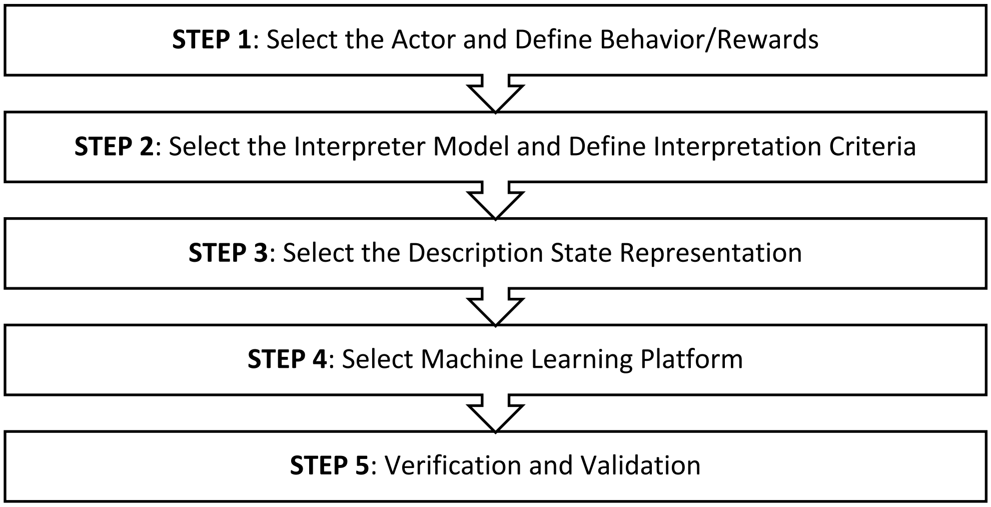

STEP 1: Select the Actor and Define Behavior/Rewards
STEP 2: Select the Interpreter Model and Define Interpretation Criteria
STEP 3: Select the Description State Representation
STEP 4: Select Machine Learning Platform
STEP 5: Verification and Validation

Figure 12: LAISR Design, Development, and Analysis Methodology

The methodology for developing and implementing a LAISR model is based on the steps outlined in Figure 12. One must first select the representation of the Actor, Interpreter, and State Representation; Once the characteristics are chosen, the Actor must be endowed with a reward system to provide a basis for how it selects actions given its current state and potential for future rewards. Consequently, the Interpreter must be selected to address how the Actor's behaviors can be interpreted. Interpretation decomposition provides knowledge of how the Actor behaved within the simulation. Finally, a State Representation is selected and employed in order to generate a *behavior narrative* for the observer. Although there are several ways to approach the implementations of a LAISR model, this dissertation focuses primarily on the use of DRL (see section 3.3) techniques for the Actor and AI interpretation techniques (0) for the Interpreter. DRLs (see section 3.3) are unique in their ability to act as complex function approximators, which have been shown to produce sometimes better than human strategies for gameplay (Beattie et al., 2016). Equivalently, AI interpretation techniques (see section 7.1) are unique in their ability to extract feature information from DRL systems.

4.6 STEP 1: Select the Actor Model, Define Behaviors and Rewards

In order to build the LAISR model from a mathematical foundation, some background in AI and machine learning theory must be introduced. This lays the foundation for the type of Actor model development that may suit the desires of the researcher.

4.7 STEP 1a: Selecting Reward Signals

One of the more challenging aspects of developing a DRL model is determining how to reward its actions to achieve maximum future rewards. With this in mind, rewards should be simple (parsimonious) and testable. And once implemented, reward systems should be verified and validated through proper code analysis and examination of results of implementing rewards. This section discusses approaches that demonstrate how to apply these techniques.

4.7.1 Parsimony: Reducing Reward Signal Complexity

Parsimony, or "just enough but no less", ensures that causal explanations and experimental descriptions contain a minimal number of factors which are essential for an explanation, understanding, and sometimes prediction (Cioffi-Revilla, 2014). With parsimony in mind, we must balance *simplicity* with *design realism* to ensure that the model maintains empirical accuracy and sufficiency. In the context of ABMs, parsimony is generally related to coded behavior, which satisfices the design of the model. On the other hand, reinforcement learning models must consider reward signals from a parsimonious perspective.

Shelton (2000) noted that often rewards could be difficult to track when combined with other signals that do not correlate to similar behaviors. Also, reward signals become even more complex within multi-agent scenarios. Leibo et al. (2017) developed methods to reduce complexity in reward signals using the Sequential Social Dilemma (SSD) approach. SSDs are partially observable multi-agent games (see section 3.2.1.1) where a single agent can obtain a greater reward displaying short term non-cooperative behavior, but the accrued payoff is higher if agents are cooperative overall. Janssen (2012) provided a method for creating a simplified approach to categorizing and implementing reward signals in terms of When, What, and Magnitude (How Much?) (See Table 6).

Table 6: Janssen Reward Structure (Source: Janssen, 2012)

Category	Description

When:	Model modifies three reward-based parameters: moment, objective function, and magnitude. Moment refers to feedback on how performance is experienced (or given).
What:	What refers to the objective function which dictates how performance is rewarded. For the objective accuracy function, each model is reinforced to reinforce accuracy of each item encoded and placed during a round of strategy.
Magnitude—how much?	The third parameter is reward magnitude. This is dependent on the objective function (OF). The OF dictates the rewarded, and magnitude states the amount of the reward. This allows the model to distinguish between different levels of success and failure

4.7.1 Reward Verification

Bastani et al. (2018) developed a process for designing rewards that could be proficiently verified. They took the approach of developing learning decision tree policies for two reasons: a) they are nonparametric and can represent complex policies, and b) they are well-structured, ensuring easier verification. The approach presented in this dissertation considered several aspects of the Bastani model, where, when developing rewards signals in code for each agent, each reward was tested to examine how it affected agent responses. In accordance with Crooks et al. (2018) model, the goal was to generate a pattern that can be reproduced and for which observation data exist. First, a set of assumptions were put in place that described the reward and generally expected results of the reward. Second, each model was trained with reward signals individually and with variations in the reward signal. Then a series of simulated experiments were tested to ensure there was a clear correlation to how rewards affected agent decisions.

4.7.2 Reward Signal Parameter Sensitivity Analysis (RS-PSA):

A parameter sensitivity analysis is the most extensively employed method for testing simulation stability (Crooks et al., 2018). The goal was to, through a quantitative measure, examine the effect that small adjustments in reward parameter values have on given model output. The approach was designed to isolate a single parameter at a time, known as the one factor at a time approach, with the remaining parameters and conditions held constant. Referring to the reward signals developed for the wargaming model (see section 5.5), a target reward was selected, and a distribution of values was tested around the reward signal to determine if the reward signal produced the expected behaviors (see Table 7).

Table 7: Reward Signal and Signal Distribution

Rule Reference: Destroy Ground Target		
	Acquired Target	Missed Target
Reward Signal	**0.2f**	**-.05**
Distribution (lower-upper bounds)	[0, .4]	[-.1.05, .1.05]

Often reward issues may arise when rewards are set too high or low, or when a reward is sparse (Pathak, 2017). It was also decided to minimize the number of possible rewards available to the agent to ensure actions were clearly related to how a reward produced an identifiable behavior related to said reward.

4.8 The Interpreter

This section introduces the most novel aspect of the model, the *Interpreter*. The Interpreter's role is to look for forms of consistent behaviors that can be termed 'rules of thumb' (Kahneman, 2011) within the RL agent model results. The Interpreter poses a substantial challenge which is, drawing the bridge between what the agent is doing (interpretability) and determining the type of behavioral strategy the agent may be using to generate its own 'rules of thumb'. Interpretability can be defined in two primary ways as it relates to human behavior and the social sciences. Miller (2019) states the following:

- **Interpretability is the "degree to which a human can understand the cause of a decision."** Here, the greater the ability to interpret a machine learning model, the easier it is to understand or comprehend why machine learning predictions have been made.
- **Interpretability is the "degree to which a human can consistently predict the model's result**." Although the term explainability and interpretability are sometimes used interchangeably, this chapter refers to the term explainability as being able to explain AI-Interpretation model predictions.

4.9 STEP 2: Selecting the Interpreter

The Interpreter is designed to construct decision policies using available data and appropriate XAI algorithms for the data. Additionally, the type of information is important when selecting the Interpreter, including whether it is primarily temporal, causal, or pattern identification matching. Additionally, the researcher may be interested in local (individual features describing the whole) or global features that aggregate features into a single correlation plot. A sampling of important XAI techniques is described in the sections below.

4.10 STEP 3: Selecting the Description State Representation

Once data is interpreted, it must be then presented in a format that can be represented as a strategy. There is not a substantial amount of literature related to the state representation of heuristics; however, a clear attribute that can help in a decomposition process is the Fast-and-Frugal Tree (FFT) and Finite State Machine (FSM) representation.

4.10.1 State Representation Level of Decomposition

One may ask at what level of state decomposition is necessary to describe a strategy? Abel (2019) provides some insight into how this question can be posed, specifically where each sub-node is primitive, and each primitive is defined by how it attempts to solve a particular problem.

Table 8: Methods of Framing State Level Decomposition (Abel, 2019)

Bounds	Description
Lower	What is the minimum number of primitive moves needed to solve a given problem?
Average	On average (across problem instances), what is the number of primitive moves needed to solve a given problem?
Upper	After how many primitive moves can we guarantee that we solve any instance of the problem?

Abel (2019) also notes that actions, or how to break down the specifics of a plan, the agent must consider resolution overgeneralization. For example, if the agent reasons over all possible permutations, then the possible paths into the future create intractability. However, if an agent makes decisions that balance between accuracy and speed, the agent is able to search over optimal actions. In other words, we seek fast-acting, utility-maximizing, and robust generalization across different state-action pairs. Unfortunately, tradeoffs exist between these variables, as shown in Figure 13.

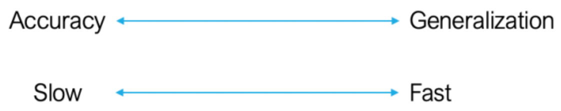


Figure 13: Considerations When Decomposing States

4.11 STEP 4: Selecting a DRL Framework

Given there are a large number of machine learning agent environments available for testing and research, several platforms are discussed. Generally, it is prudent to focus on the development of the process for building scenarios and rewards rather than developing the code to implement specific DRL code. Example variables may also be necessary to consider when selecting a DRL agent framework:

- State of the art RL algorithms implemented
- Documentation/tutorials and examples
- Robust cross-platform support
- Open-source code that's easily modifiable
- Regular updates and an active community

- Quality visualization

The following is a non-exhaustive list of deep reinforcement learning models currently in use in the machine learning community for just such efforts.

4.11.1 Unity Platform

Unity is a framework for game development that supports several core areas; importing art and assets, 2D and 3D, modeling; assemble assets into scenes and environments; audio, 3D models, physics and animation, AI interactivity, and gameplay logic; and edit, debug and optimize the content for your target platforms (Juliani, 2018).

4.11.2 Arcade Learning Environment

The Arcade Learning Environment (ALE) is a Deep Q-Network (DRL) based system designed to achieve expert-level competency on Atari console games (Mnih et al., 2015; Bellemare et al., 2017).

4.11.3 DeepMind Lab

DeepMind Lab (Lab), derived from the Quake III game engine, has been used extensively by the DeepMind Lab for researching reinforcement learning systems (Beattie et al., 2016). Given the nature of the engine, simulations are designed primarily as first-person, which may not be relevant to strategy and multi-agent models (Juliani, 2018).

4.11.4 Project Malmo

Project Malmo is a Minecraft game-based platform (Johnson et al., 2016). Although limited in flexibility (issues of low resolution tied to game type), several research projects have been developed in the environment (Oh et al., 2016; Shu &Socher, 2017; Tessler et al., 2017).

4.11.5 VizDoom:

Kempka et al. (2016) generated VizDoom (based on the game Doom) as an early first-person Deep Reinforcement Learning framework (Kempka et al., 2016). Some work in learning curricula (Wu & Tian, 2016) and memory (Lample & Chaplot, 2016) have been included in the framework, although it is generally considered lower fidelity than Deep Mind and Unity.

4.12 STEP 5: Verification and Validation Process

For each representation of the LAISR Actor Model (as well as Interpreter (see section 4.8)), a verification and validation process should be followed. Robust training and testing are implicit in the development of accurate models. However, it is difficult to guarantee that a system returns an expected value given the “black box” attributes of the neural network structure. In large and complex models, computing all possible outputs for a given set of inputs is intractable due to the number of potential discrete or continuous states. However, when training a model, one can first develop efficient methods to test inputs and outputs.   Many properties that cannot be verified offline can be verified at runtime, although this might not always be feasible with regards to computation time or resource efficiency. Instead of verifying the entire specification, only the affected parts can be verified at runtime, assuming that if the specification was verified at the start of learning, and each change is deemed valid, then the specification is still valid after an arbitrary number of changes.

4.12.1 Verifying Code Process

Verification processes included a formal review process for code to include its comparison to several code examples provided by all code libraries. A second coder provided some experience with machine learning and generated a set of concepts that could extend existing examples. Within this framework, each concept was generated in a pair/review programming review where one coder wrote the code, and another would read it. Additional comments were included for each aspect of the model, as this allows the model design to be easily explained to others in the future.   A beta test for each simulation was run where the model developer provided conditions for running the model (test cases), and results were evaluated based on the test cases.

4.12.2 Model Validation

Validation is an interesting problem related to reinforcement learning, given we cannot always be sure that the agent produced an optimal policy. In reinforcement learning, regret$R$ (François-Lavet, 2018) is a very commonly used metric where, at each time step, one takes the difference between the reward of the optimal decision versus the decision the algorithm actually took. We could then sum each$R$ for cumulative regret. As each agent performed its optimal policy, the smaller the R, the better an algorithm has performed. The difficulty with the regret approach to validation is that it assumes there is a single optimal decision policy. This may be difficult if more than one policy is determined. The work presented in this dissertation dealt with this issue by ensuring that all agents of a certain type within a single simulation shared a collective neural network. This would ensure a single optimal decision was selected.

4.13 Challenges Verification and Validation Challenges of ABMs

Although section 1.1 outlined several challenges related to the design of a proposed boundedly-rational agent, there are several other criticisms that relate to verification and validation challenges. These critiques have to do with dealing with large, parameter intensive models and human bias.

4.13.1 The "Curse of Dimensionality"

ABMs simulate the evolution of complex systems with a set of parameters without fully knowing the optimal parameter states to describe the real-world environment (Li et al., 2013). These parameters provide a rich expressiveness, which provides a broad state space for examining a complex domain. Although with this rich set of parameters comes the "curse of dimensionality" (Busoniu, 2010) that leads to an exponential number of critical points along with the parameter space, with multiple local maxima, minima, and saddle points, which negatively impact the performance of gradient-based search procedures (Wong, 2015). Even for small models, exploring the behavior of the model through all possible parameter combinations (a full factorial exploration) is practically impossible, even employing multi-objective optimization procedures such as multimodal optimization or niching (Li et al., 2013). Parameter-intensive systems suffer from the unpredictability of the results due to dependencies between parameters (Alahi et al., 2016). Agent models also develop complex state spaces with a very large result space. In practical terms, integrating a large number of details into a model will make generating agent-based models difficult to evaluate, as each feature within the model needs to be defined and integrated with the other model components in a meaningful way. Within this so-called "curse of dimensionality," increasing the variables integrated with the model increase the potential results. If we want to obtain meaningful statistical results, it is useful to either keep the number of variables as low as possible or increase the number of agents and runs. Although this is a design challenge, it also relates back to section 1.1.1, demonstrating that an agent could, through its own sub-systems, minimize the amount of complexity and dimensionality that is in the hands of the model developer.

4.13.2 Human-Intervention Bias

Currently, there is no true automated method for developing agents that can produce desired results without direct human intervention. Thus, at any point in the development of a complex model, even if the verification method was done with great scrutiny, any number of parts of the system could generate errors which will then be accumulated within the larger system. Ideally, a system would be able to adapt to its conditions and make its own best model without developing specific 'pre-designed' rules. Modelers will often provide agents in an ABM with discrete rules that control how the agent behaves in response to its local environment. These behavioral models are intended to be reasonable estimates of real-world decision-making. Yet human-biased representations may limit opportunities for an agent to build their own utility-maximizing, decision-making representation of its environment (Osoba et al., 2020). This inherent design challenge is also discussed in section 1.1.2 where agents do not build their own heuristics.

4.13.3 Validating Behaviors

In order to design a self-organizing system with the desired emergent behavior, it is important to find local rules for the behavior of the system's components (agents) that generate the intended behavior at the system scale. In many cases, this is done by a trial-and-error process, which in the case of systems with high complexity, is not efficient or even unfeasible. Agent models also suffer from the unpredictability of the results due to unexpected dependencies between parameters. This gap addresses the general concern that agents must be predictable in their behaviors, given they are designed with specific actions in mind. This is an understandable challenge and will not easily be removed from standard ABM design. However, the proposed model does not assume that it has a specific behavior in mind. Rather, the hope is that the agent will help us uncover strategies to optimize its behaviors.

5 LAISR EXPERIMENTS: HOMOGENEOUS MODELS

This chapter presents two LAISR experiments, the Schelling (see section 5.1) and a Tactical Warfare (see section 5.5) experiment. The primary purpose of this chapter highlights the *Actor* implementation, where I leave the more complex Interpretation implementation until *Chapter* 7*: LAISR Experiments: Advanced Explainable Artificial Intelligence* .

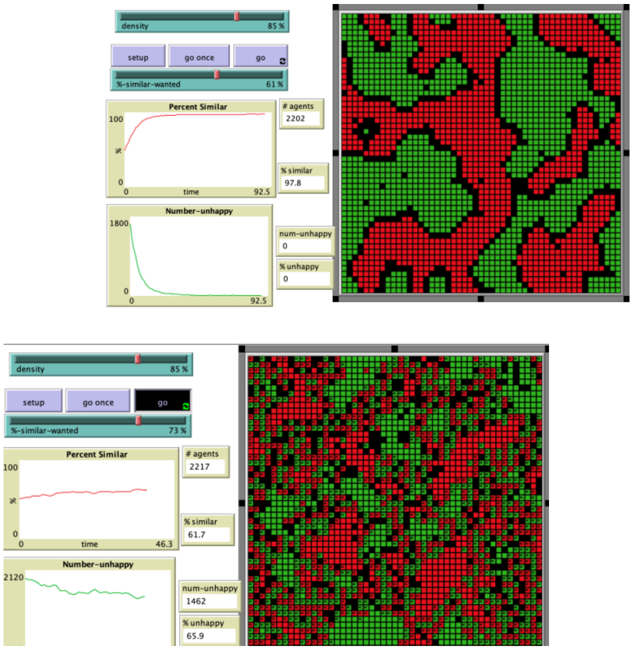
5.1 Experiment 1: Schelling Experiment

The first experiment is intended to provide an existing example of how the LAISR model can be applied to social science theory. This example demonstrates the simple yet well-known Schelling segregation model (Schelling, 1971). The Schelling model of segregation is an agent-based model that illustrates how individual tendencies regarding neighbors can lead to segregation. The model is particularly beneficial for the study of segregation of ethnic groups where agents represent householders who relocate in the city (Ibid). Within the model, each agent is part of one of two groups and aims to reside within a neighborhood where the fraction of similar agents is satisfactorily high: above a predefined tolerance threshold value F. It is known that depending on F, for groups of equal size, Schelling's residential pattern come together as to either complete integration (a random-like pattern) or segregation.

Table 9: NetLogo Schelling Model Parameters

Parameter	Type	Description
DENSITY	Variable	The slider controls the occupancy density of the neighborhood (and thus the total number of agents).

The %-SIMILAR-WANTED	Variable	The slider controls the percentage of same-color agents that each agent wants among its neighbors. For example, if the slider is set at 30, each green agent wants at least 30% of its neighbors to be green agents.
The % SIMILAR	Monitor Output	The monitor shows the average percentage of same-color neighbors for each agent. It starts at about 50% since each agent starts (on average) with an equal number of red and green agents as neighbors.
The NUM-UNHAPPY	Monitor Output	The monitor shows the number of unhappy agents, and the % UNHAPPY monitor shows the percent of agents that have fewer same-color neighbors than they want (and thus want to move). The % SIMILAR and the NUM-UNHAPPY monitors are also plotted.
The VISUALIZATION	Combo Box	The combo box gives two options for visualizing the agents. The OLD option uses the visualization that was used by the segregation model in the past. The SQUARE-X option visualizes the agents as squares. The agents have X's in them if they are unhappy.


density 85 %
%-similar-wanted 61 %
# agents 2202
% similar 97.8
num-unhappy 0
% unhappy 0

density 85 %
%-similar-wanted 73 %
# agents 2217
% similar 61.7
num-unhappy 1462
% unhappy 65.9

Figure 14: NetLogo Schelling Model (Source: Wilensky, 1997)

In the NetLogo Model (see Figure 14: NetLogo Schelling Model), agent and environmental parameters are set, and the observer can view how parameter changes can affect the results of the model, both in the amount of segregation and the time it takes to achieve a stable state.

5.2 Schelling LAISR Model Development

The Schelling LAISR model (see Table 10) employs a DRL Q-Learning (see section 3.2.4.1) and a set of rewards necessary to help the agent find optimal actions when in specific states.

Table 10: LAISR Methodology for Schelling Reinforcement Learning Model

SELECTION STEP	SELECTION
Step 1: Select the Actor and Define Behavior/Rewards	Q-Learning Model (see section 3.2.4.1) and Reward Signals (See Table 17)
Step 2: Select the Interpreter Model and Select Interpretation Requirements	Heat Map Representation (See **Figure 17: Heat map representations of actor**)
Step 3: Select the Description State Representation	Not Implemented in this Model
Step 4: Select Machine Learning Platform	See Sert et al. (2020) paper
Step5: Verification and Validation	See Sert et al. (2020) paper

Sert et al. (2020) developed the Schelling implementation with a set of rewards that could generally imitate the ABM model. When developing a reinforcement learning model (see section 3.1.4) compared to traditional ABMs (see section 2.1), one must first examine the existing social science model, then determine how to revise the model to support reward signals. In the Schelling NetLogo implementation (Wilensky, 1997), agent behaviors are defined purely by parameters. The model rewards, R, are scalar values that are given to each agent as it completes an action at a given state. The final summary reward for the agent is the sum of rewards based on the signals in Table 11.

Table 11: Reward Table for Schelling Model (Source: Sert et al. (2020))

Reward Signal	Description
Segregation reward (SR).	This reward promotes agents' segregation, in the form: $SR=s-\alpha d$SR=s−αd, where s is the number of agents of a similar kind within the agent's observation window, d is the number of agents of different kind and $\alpha\in[0,1]$α∈[0,1] is a parameter used to control the intolerance of agents to be next to those that are different from them.
Interdependence reward (IR).	This reward promotes interactions among agents of different kinds. When an agent meets another agent of different backgrounds, a winner is selected, i.e., one who moves to the cell occupied by the other agent. The winner receives a positive reward and a life extension of one iteration. The loser dies.
Vigilance reward (VR).	This reward reinforces agents that stay alive $VR=0.1$reward for every time step they survive and $VR=0$ when they die.
Death reward (DR).	Agents are punished (or killed) who lose interactions against agents of the opposite agent type. Agents receive $DR=-1$reward when they die or $DR=0$ when they stay alive. Agents must learn that being killed is not rewarded, primarily to reward non-risky behavior.
Occlusion reward (OC).	This reward punishes movements into occupied cells between agents of the same kind. If an agent tries to move towards an area occupied by an agent of its own kind, the agent receives $OC=-1$reward. If the agent moves towards a free cell, it receives $OC=0$.
Stillness reward (TR).	Signal promotes the exploration of space by punishing immobility. Agents who remain immobile receive $TR=-1$reward. Agents who chose to move receive $TR=0$.

5.3 Results and Interpretation: DRL Schelling Experiment

Figure 15 displays agent behaviors for multiple values of aggregated rewards (rows) and times (columns). Rows demonstrate results aligned to the distinctive values of the interdependence reward (IR). Columns demonstrate the states at differing times within the simulation. Heat maps are achieved by generating a mean across 1000 iterations. Panel (a) red regions signify "biased occupation of type A" agents are fully occupied with type A (agents are a concentration of values of +1). Blue regions signify the occupation of B agents, and full blue occupation is -1. White areas indicate across the mixing of types and are denoted as concentration 0.

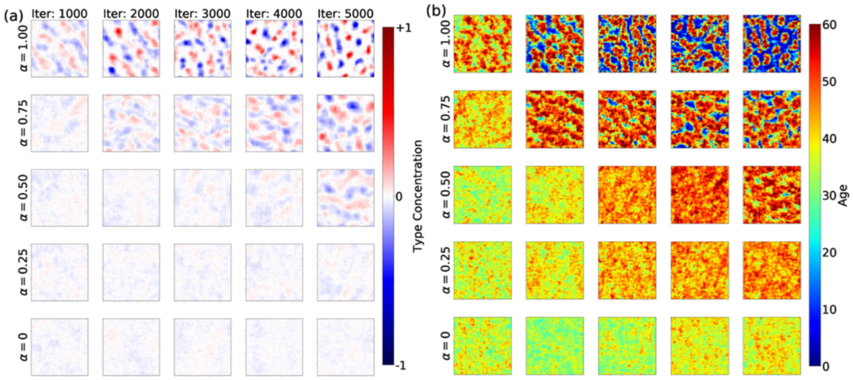


Figure 15: Results of DRL Schelling Experiment (Source, Sert et al. 2020)

In Panel (b), color signifies the age of agents regardless of type. As color changes from blue to red, agent age rises. The heat maps are presented over a single trial of the experiment. The dynamics of segregation rapidly-produce patches of segregated types (top panels). As interdependence rewards increase, the probability of one grid being occupied by an agent of type A or B becomes more alike, and plots are represented whiter (bottom right panels). By creating interdependencies among agents, they increase their interactions and reduce spatial segregation. Sert et al. note that increasing the connection of rewards diminishes spatial segregation among different types. In sum, this study succinctly demonstrates how reward signals using reinforcement learning can be applied to social science models effectively.

5.4 Experiment 2: Demonstrating LAISR with General Interpretability

A second, more complex LAISR example was developed with the intention of demonstrating the concept of the Advantage Actor-Critic (see section 3.2.5.1) DRL method. The Advantage Actor-Critic model was selected as it has proven more precise with curiosity modeling as compared to the DQN approach within the Unity platform (Juliani, 2017). The model also includes a simplified Interpreter, which generates a statistical translation of output results towards general behavior and strategies. Finally, the State Representation uses a Finite State Machine (see section 2.5.2.2) representation for readability and a path to code representations of the agent strategies.

5.5 Tactical Warfare Concept

Table 12: LAISR Methodology for Homogeneous Multi-Agent Reinforcement Learning (Tactical Warfare)

SELECTION STEP	SELECTION
Step 1: Select the Actor and Define Behavior/Rewards	Advantage Actor-Critic DRL Model (see section 3.2.5.1) and Reward Signals (Table 14: AC3 (P,A,R) definitions and rewards)
Step 2: Select the Interpreter Model and Select Interpretation Requirements	Statistical Representation and Heat Map Description
Step 3: Select the Description State Representation	Finite State Machine Representation
Step 4: Select Machine Learning Platform	Unity ML-Agents
Step 5: Verification and Validation	See Section 4.12

Table 13: Code Link for LAISR Methodology (Tactical Warfare)

CODE LINK	DESCRIPTION
https://github.com/paulsimvient/Homogeneous-MultiAgent	Tactical Warfare Concept

Tactical Air and ground warfare are complex; it involves many dimensions, complicated processes, high costs, and significant hazards, and its doctrine is a set of specialized knowledge on the execution of combat maneuvers (Yining and Yuxian, 2003). The same doctrine is used to generate discrete rules within a commercial-grade simulation platform using computer-generated force (CGF) agents. This is based on a traditional 1-v-1 pursuit-evasion problem in three-dimensional airspace (Ardema, 1985). A wargame was developed as a simulated, two-sided (Blue and Red) game, where the operation is modeled in a game-based environment. Referring to the LAISR model, the intention was to first generate a set of RL models that provide a type of optimal decision-making, and second, create a means to derive descriptive behavior representations in readable formats. Within the simulation, there are three layers; a) a *Game/Simulation Layer* that represents the simulation content and structure of the overall simulation design, b) a *DRL Layer* where agents generate rewards for achieving mission goals, and c) the *Interpretation Layer* demonstrates the decomposition process of the agent's policy into a set of finite state based models for evaluation.

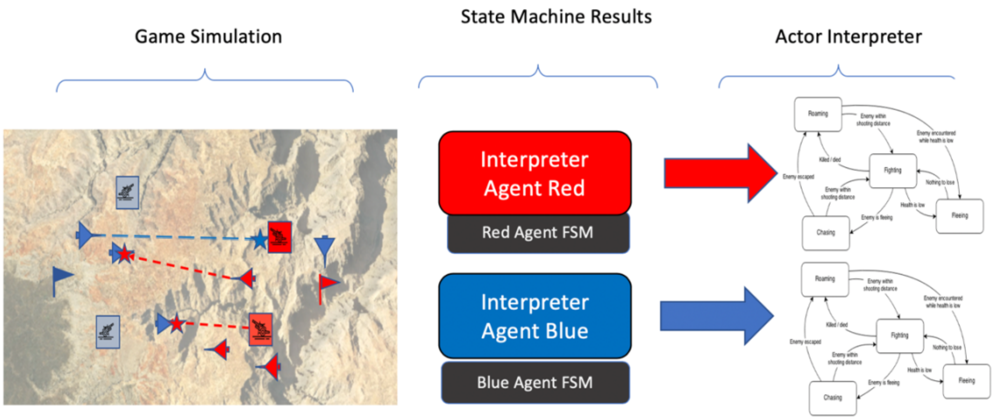


Figure 16: LAISR Simulation Example

5.5.1 Unity Platform Selection

A Unity-based simulation was selected to provide the essential characteristics of tactical air warfare operations (See 4.11.1). Unity3D was selected for several reasons; One, it is a free game platform and toolkit which has been designed to research agent models using several reinforcement learning methods; 2) Within the platform, multi-agent interaction, and Unity agents are trained using Google's TensorFlow and Keras packages (Juliani, 2018) which are both well-adopted frameworks for machine learning; and 3) Unity provided several examples that allowed experimentation with temporal domains (e.g., long short term memory), curiosity and sparse rewards, and competitive multi-agent machine learning.

The model was designed based on a set of general criteria necessary to begin the wargaming experiment. A common understanding of the wargaming objective was created, including the reason for running the scenario, learning objectives, and external conditions and limitations. A simulated area of 10 km x 10 km area within the game environment was developed to be observable from both a top-down (command and control) and a 3D man-in-the-loop (MITL) perspective. Within this model, there were 20 LAISR Agents - ten red and ten blue force agents - in the simulated three-dimensional environment. All agents were tasked to outmaneuver each other so as to enter into a favorable position to eliminate each other using missiles. The objective of the model was to develop opposing agents which learn intrinsically

a) that each red/blue agent must first destroy the ground weapon, and

b) once ground targets are destroyed, engage air targets while not being shot down.

The winning team had the most agents in the air at the end of the episode. The game layer was developed using the Unity ML-Agent extension library (Juliani, 2016) to develop both the *Actor* and the *Interpreter*.

5.6 Actor Agent Design

For the Actor Agent, a reward system as shown in Table 14 was developed within a 3-tuple: case = (P, A, R) where: P is the description of the action, containing pertinent information about the state of the agent $(a\ state\ s\ \in\ S)$; A is an action (or a sequence of actions) that must be performed for the problem at hand; R is the expected reward for performing the action (Ros, 2009). For this simulation, ten agents contribute to the same continuous action space. The game rules are as follows:

- Rule 1: Destroy as many enemy agents as quickly possible.

- Rule 2: Prioritize destruction of ground weapons: Ground weapons should be destroyed first to minimize risk to a fighter squadron.
- Rule 3: Once fighters are safe from ground weapons, destroy enemy fighters
- Rule 4: At any cost, do not get shot down

Each of the Red and Blue force agents, in this case, the stealth tactical fighters, are given reward signals for completing a number of actions with no specific tactic for how to do so. Reward signals (see Table 14) are used to train the agent's optimal policy.

Table 14: AC3 (P,A,R) definitions and rewards

Rule Reference	P	A	R	
			Acquired Target	Missed Target
Rule 1 & Rule 2	A.0	Destroy ground targets	0.2f	-.05
Rule 1 & Rule 3	A.1	Destroy air targets	.12	-.01
Rule 1	A.2	Temporal Penalty	-0.1	
Rule 4	A.3	Shot Down	-5	

5.7 Actor Results

Experiments were conducted by generating the conditions for developing Actor agents that used an intrinsic curiosity model as a means to delineate two primary behaviors; one, target and destroy ground weapons, and two, engage enemy fighters. During each run of the simulation, Actor agents explore the action space and respond based on reward incentives from Table 14. The Actor agents were trained over 500,000 episodes within the Unity ML-Agents (see section 4.11.1) environment using the Advantage Actor-Critic model (see section 3.2.5.1) model. The results were evaluated using the TensorFlow analytics website Tensorboard which demonstrates the speed and accuracy of model training. The model is able to achieve stable rewards, taking approximately 12 hours on an NVIDIA GeForce RTX 2060 6GB. Reviewing 500,000 episodes, Table 15reveals the cumulative reward for the agent training increased over the episodes as well as minimum and maximum rewards over the 500,000 episodes.

Table 15: Actor agent results: Cumulative reward

Reward Type	Value
Result reward	86.3387494
min reward	-21.9573917
Max reward	176.8109988
Count (episodes)	500000

Once the Actor agents were trained, 500 skirmishes were run between the red and blue teams. A heat map was created that showed locations of red and blue fighters. **Figure 17**presents data for both red and blue Actor agents. Column (A) presents locations before the ground weapon is destroyed. We find that in this column, opposing agents converge on the general artillery location. In Column (B), we observe agent behavior *after* the artillery is destroyed. The heat maps are generated over 1000 iterations, where the red regions demonstrate significant occupation of agents. Light blue regions denote some occupation of agents where full blue is indicated by no concentration. A white cross denotes the location of the ground weapon before it is destroyed. The results of the skirmishes showed agents converging towards enemy ground weapons until they are destroyed, then they engage in air-to-air combat.

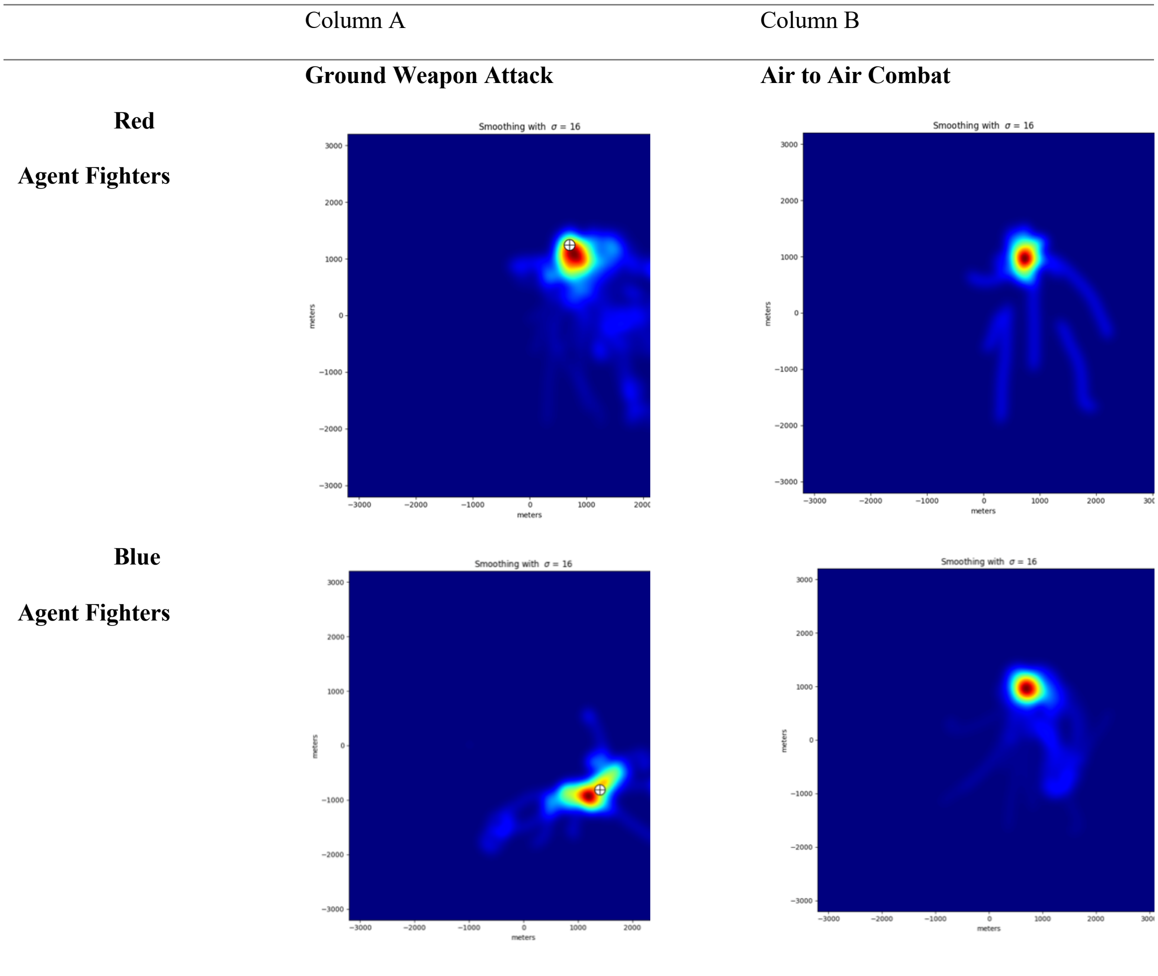
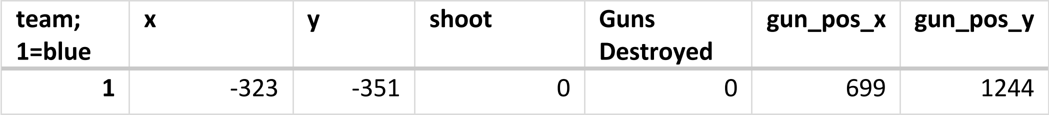


Figure 17: Heat map representations of actor agent location in a metered grid

5.8 Interpreter Agent Results

This section demonstrates a simple example of an interpretation model. It is important to note that a full AI interpretation model is discussed in *chapter 7* although this early example provides some methods that demonstrate interpretation. Referring back to the Interpreter, the goal is to uncover behaviors that can be considered strategies. ~500 agent trials were run within the simulated environment, given each of the fighter agents interacted with ten adversarial entities and nine friendly entities. All agent actions were stored in CSV tables (see Table 16) with 8000-20000 entries total depending on the length of the skirmish, across the red and blue agents as strategies A.0-A.3 (see Table 17). The CSV file stored agent type (0=red, 1=blue), x and y positions, whether it was shooting, if the ground weapon was destroyed, and the ground weapon and y position.

Table 16: CSV Format for Homogenous Agent Skirmish



team; 1=blue	x	y	shoot	Guns Destroyed	gun_pos_x	gun_pos_y
1	-323	-351	0	0	699	1244

Table 17: Actor-Interpreter dominant use of strategy

Strategies		Interpreter	
		Pre-Ground Weapon	Post-Ground Weapon
A.0	Destroy ground targets	56.30%	4.21%
A.1	Destroy air targets	2.45%	60.00%
A.2	Observe the ground target current state	0.03%	3.00%
A.3	Observe locations of hostile	14.00%	4.60%

Table 17shows the results of how the Interpreter designated Actor behaviors. One could observe that both the Interpreter observed a strategy for spending time destroying the ground weapon; once the ground weapon was destroyed, agents changed their dominant strategy to air-to-air combat. These calculations were then used to develop a FSM with probabilities of entering each of these states.

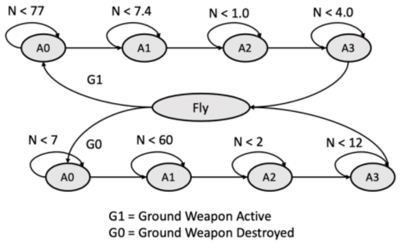


Figure 18: Actor Finite State Machine

In Figure 18, the plane stays in a fly state when the ground weapon is active (G1). While in G1, each sub-state is iterated through, dominant states are given more time and therefore are more likely to be activated by the FSM. Moving into the ground weapon destroyed state (G0) produces a second set of dominant states with differing levels of priority.

5.9 Summary

In this chapter, the dissertation introduced a method that demonstrates how machine learning can be used to model decisions made by agent models. The decomposition of neural networks for mission planners, instructional designers, and agent modelers, amongst others, can provide a significant way to find optimal solutions to complex scenarios. The approach also provides the basis for how this type of decomposition can be used to develop DRL-Agents (see section 3.3) to refine our ability to solve and manage agent behaviors. The initial results of the Interpreter (see section 4.3) model demonstrate promise as a mechanism to derive strategies. Using these models to provide results to problem-solving and reasoning can help learners transfer and apply their knowledge to novel problems and situations (Rudd, 2010). These models also provide methods to generalize across operating environments, adversary, and even game/simulation tools.

6 LAISR EXPERIMENTS: HETEROGENEOUSMODEL

Until now, the dissertation has discussed the concept of the LAISR model (see section 3.3) as primarily homogeneous entities that interact against one another. Epstein (2006) notes that heterogeneous agent populations change or adapt endogenously over time. This next phase attempts to generate a set of heterogeneous multi-agent reinforcement learning models (see section 6.1). This section introduces a heterogeneous multi-agent DRL (see section 3.3) model. The LAISR-Actor is designed to work in cooperative, fully competitive, and mixed environments (Zhang, 2019). This chapter investigates the challenges of coordinated learning across heterogeneous agents.

This section also addresses a key concern in developing heterogeneous multi-agent models, the problem of *nonstationarity* (see section 6.1.1). Nonstationarity becomes problematic where the Markov property (see section 6.1.3) assumes agents of different types must contend with a continuously changing environment.

6.1 Multi-Agent Systems (MAS) in Reinforcement Learning

Multi-Agent Systems (MAS) can be described as two or more agents interacting with each other in a common environment that acts in response to individual goals (Busoniu et al., 2008). That is, in lieu of having a centralized singular model, each agent plays a role in decision-making (Ibid). Multi-agent reinforcement learning (MARL) is an extension of single-agent reinforcement but must contend with additional areas of complexity. Primarily, heterogeneous goals among agents (Agogino and Tumer, 2005;

Busoniu et al., 2008) and multiple agent parameters (Panait and Luke, 2005), and scalability (Busoniu et al., 2008).

6.1.1 Problems of Nonstationarity

A challenge with Heterogeneous Reinforcement Learning-based Multi-Agents, as described by Castaneda (2016), is *nonstationarity*, which occurs because the interaction of multiple agents constantly reshapes the domain space. Where in single-agent RL, the agent is observing only the effect of its own actions. In MARL, agents are interacting and learning concurrently, and agents must associate an action to certain outcomes as well as to another agents' behavior. Nonstationarity is a fundamental problem in traditional cooperative MARL whereas each agent relearns other agent policies; this causes information convergence to be slow (Papoudakis et al., 2019; Hernandez-Leal et al., 2017).

6.1.2 Example: Bat/Frog System Revisited

Consider the following: an environment has two agents discussed early in the *section Given the social* science community's interest in modeling decision processes, ideally, we would like to be able to provide techniques that mimic human cognitive processes, i.e., quick decisions, particularly when working with complex data. Although these 'heuristics' made may not necessarily be optimal, they can aid our understanding of how humans acquire and employ decision strategies.

An Example ABM to Illustrate the Challenges: The Bat-Frog Predation Model (0), where a frog and bat must learn to exist together. Frog's policy must have knowledge of bat's policy, which from its perspective is a part of the environment (and the opposite is true for bat's policy). At each step of learning, the frog learns about the bat's policy and its environment. Bat then learns about the environment and the frog's policy, updating his policy and making the frog's knowledge of his sightly wrong. Now frog must learn bat's new policy and update its own, making the bat's knowledge slightly wrong. This *ringing* of information can greatly slow convergence during learning, especially for highly coordinated tasks with many agents, and this specific form of nonstationarity is believed to be a fundamental reason why it's so difficult to converge to good policies in multi-agent learning Papoudakis et al. (2019). Therefore, it is one intention of this dissertation to develop a convergence approach where both parties are generally satisfied with their policy. The reader is reminded that there is an important differentiation that exists between the homogeneous model and the heterogeneous one.

- Homogeneous models maximize their own current and future rewards with no bearing on any other considerations
- Often (but not always), heterogeneous agents use game-theoretic methods to generate stable states between themselves and another agent type in the environment. For example, our bat will not achieve its rewards if it kills off all the frogs in the environment.

6.1.3 Multi-Agent Markov games

One approach to MARL development is the use of Markov games, where multiple adaptive agents will interact with opposing goals. This is where precisely, two agents with opposing goals share an environment. The discussion begins with the discussion of a Q-learning-like (see section 3.2.4.1) algorithm for developing policies and demonstrating its use within a two-player game in which the optimal policy is probabilistic. Unlike MDP's (see section 3.1.4.1), deterministic policies are not necessary. Instead, the policy is often probabilistic and stationary, mapping discrete states to probability distributions. The Multi-Agent Informational Learning Processes (MAILP) model, introduced by Terry and Grammel (2020), is a novel model of information transfer during multi-agent learning. They used the Multi-agent informational learning process (MAILP) to show that increasing training centralization arbitrarily mitigates the slowing of convergence due to nonstationarity. Here, the MAILP model demonstrates MARL converges slowly under normal circumstances due to nonstationarity and that centralization during learning arbitrarily improves this (with parameter sharing having the greatest level centralization). In another approach, Lowe (2017) employed a Multi-agent Actor-Critic (MAC) algorithm (see section 3.2.5.1), which gives each agent a central and global critic during the training process.

6.2 Experiment 2: The Heterogeneous Multi-Agent Reinforcement Learning LAISR Model

Table 18: LAISR Methodology for Heterogeneous Multi-Agent Reinforcement Learning

SELECTION STEP	SELECTION
Step 1: Select the Actor and Define Behavior/Rewards	Advantage Actor-Critic Model (see section 3.2.5.1) and Heterogeneous Reward Signals (Table 20). Adversarial Self-Play (see section 6.3)
Step 2: Select the Interpreter Model and Select Interpretation Requirements	Random Forest Partial Dependency and SHAP AI Interpretation (0)
Step 3: Select the Description State Representation	Finite State Machine Representation (see section 2.5.2.2)
Step 4: Select Machine Learning Platform	Unity ML-Agents (see section 5.5.1)
Step 5: Verification and Validation	See Section 4.12

Table 19: Code Link for Heterogeneous Multi-Agent Reinforcement Learning

Code Link	Description
https://github.com/paulsimvient/Sheep-Wolf	Heterogeneous Multi-Agent Reinforcement Learning

The heterogeneous model examines the equilibrium state between dissimilar agents within a population of carnivores and herbivores. Unlike the first round of homogeneous agents (see section 5.1), this experiment demonstrates how differing agent types (with different reward signals). The concept presented in the proposed model was recreated based on a Lenham (2018) *Terrarium* model, although the presented model contained its own approach towards generating the reward signals. The terrarium contains a pre-defined number of carnivores and herbivores as well as plants. The plants grow at a constant rate and can be eaten by herbivores. Additionally, plants can spread seeds and thus expand their locations of sprouting in the virtual environment. Carnivores move within the space and eat herbivores. Both herbivores and carnivores need the energy to survive and dissipate energy as they move through the environment. If a maximum energy threshold is reached, an agent can reproduce. The goal for each of the agent types is to dominate the space, i.e., no opposing member is left in the environment.

6.3 Adversarial Self Play

In a customary DRL (see section 3.3) training condition, an agent increases its reward signal towards a maximum accumulated reward. These signals are encoded as agent tasks, such as navigation, behaviors, and actions. Certain limits are applied to agent behaviors such as constrained speed, forces, physical constraints (e.g., walls), and the agent must work within these constraints while maintaining a maximized reward. Unlike standard reinforcement learning scenarios, adversarial self-play agents compete with opposing agents where, from its perspective, is effectively part of the environment. Each agent receives its own *Nash Q-Value (*Hu and Wellman, 1998*)* where Q-values now must consider cooperative actions, rather than just individual actions.

6.4 Reward Signals and Parameters

Table 20 presents the reward signals and descriptions (See Table 20) that are used to reinforce the DRL behaviors noting that the herbivore is penalized (-.25) for eating an herbivore, and carnivores are penalized for eating plants.

Table 20: Carnivore Herbivore Reward Signals

Reward	Value	Carnivore	Herbivore	Description
Reproduce	1	x	x	Reinforced to reproduce through reaching a maturity size. Creates reproduced agents
Eat Agent	.5/-.25	x		Agents contain energy, even after they die, so carnivores can eat dead herbivores. If it eats an herbivore with energy, it received a positive reward; if it eats one with no energy (dead too long), it receives a negative reward.
Eat Plant	.5	-.25	x	Herbivore eats plants
Attack Agent	.5	x	x	Rewards attacking alternate type, which dissipates a competitor’s energy
Kill Agent	1.0	x	x	If the opponent is killed, reward
Out of bounds	-1.0	x	x	If the agent is out of bounds, create a negative reward

Initial parameters were set for each of the opposing agent types - both herbivores and carnivores (See Table 21). Noting that this is a toy model, parameters were designed to generate some standardization in game playtime. In other words, herbivores were on the lookout for plants to eat, which grew at a different rate than both herbivores and carnivores. Additionally, carnivores could eat herbivores, but not vice versa.

Table 21: Agent Parameters (Herbivore/Carnivore)

Parameter	Value	Discussion
Max Energy	1	Maximum energy dissipates with time, movement, and attacks.
Mature Size	5.0	Maximum size is used to normalize the speed of growth.
Growth Rate	3.0	Value is used to increase the size of all agents, which in turn supports the amount of damage that an agent can incur. e.g., defenseDamage = Defense + (Size / 10);
Eating Speed	.3	Controls the amount of energy consumed by agents when eating.
Decay Rate	.001	Once an agent is dead, the decay rate reduces the energy of the corpse
Max Speed	2.5	Speed in movement within the simulated environment
Attack Damage	0	Amount of attack damage received by agents
Defend Damage	.5	Used as a value to calculate overall damage = AttackDamage - vic.DefendDamage;
Eyesight	20	Distance line of sight for each agent

6.5 Results: Heterogeneous Agent Model

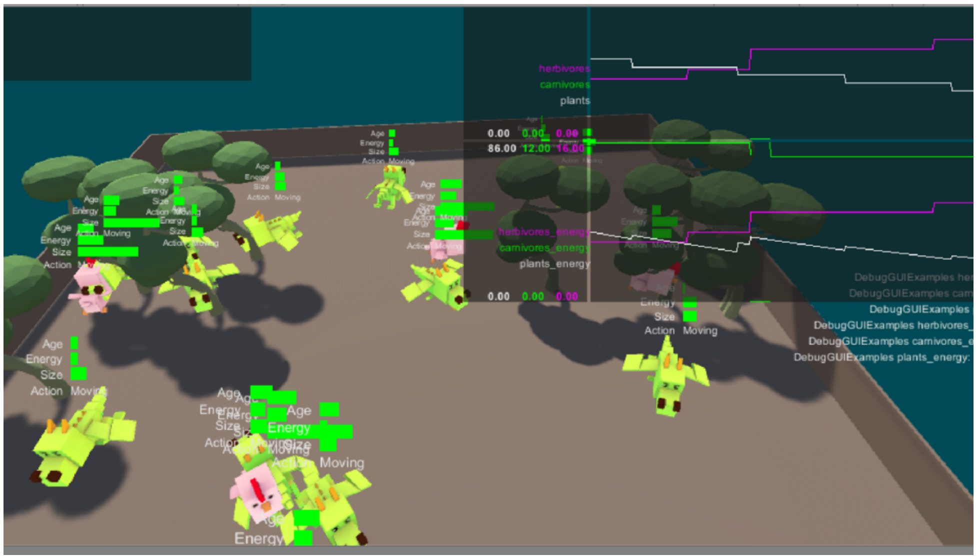


Figure 19: Wolf Sheep Model

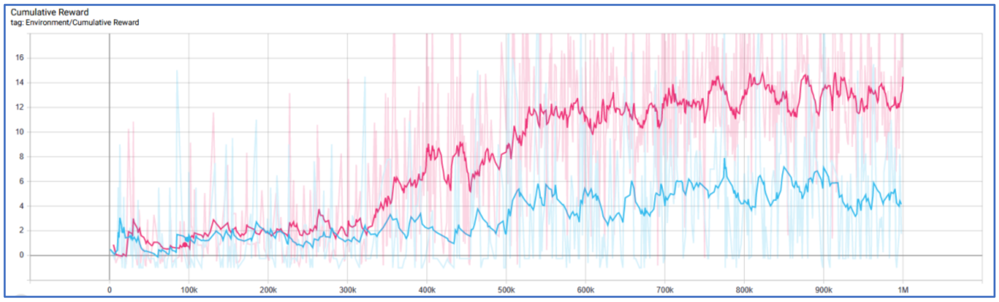


Figure 20: TensorFlow Output (Red) Carnivore (Blue)

All experiments were performed on a six-core i7 8700k @ 3.70 GHz, with an NVIDIA GTX 1080 GPU, using TensorFlow-GPU v1.7.1. The training time was roughly 4.5 hours. Results are tested on both the PC architecture, as shown above, and a MacBook Pro 2016. The herbivore agent over 1,000,000 iterations demonstrated a more optimal learning strategy over the carnivore. It is likely that herbivores may have had more players within the simulation as they generally tended to gather more energy as compared to the carnivores. It is possible with future iterations, parameters can be adjusted to minimize the growth of herbivores, but this ensures values between herbivores and carnivores were as consistent as possible. The next chapter, *LAISR Experiments: Advanced Explainable Artificial Intelligence* explains the results through advanced AI interpretation methods.

7 LAISR EXPERIMENTS: ADVANCED EXPLAINABLE ARTIFICIAL INTELLIGENCE AND STATE REPRESENTATION

Research Question RQ 2

In the previous chapter, two distinct heterogeneous DRL-Actors (see section 3.3), the wolf and sheep, were introduced. Picking up where the previous chapter left off, the concept of Explainable Artificial Intelligence (XAI) is developed into an example for the Wolf-Sheep Predation model (see section 6.2). In reference to research question two (see section 1.3), XAI techniques are implemented that are designed to decompose deep learning models. This will help to enable human users to understand, trust, and describe the emerging generation of AI algorithms, as discussed in this dissertation. In the section part of the chapter (7.10), a State Descriptor will be created, which presents XAI data in a format that describes the agent's strategy as a set of states and probabilities.

7.1 Interpreting the Heterogenous LAISR Model

The model interpretation process demonstrates ways to show prediction between actions and how these actions affect change within the model. The first objective is to identify the most significant and remove insignificant ones; this gets us to a result in much shorter training time. For model interpretation, the following steps are taken:

1) Select Algorithms
2) Display Descriptive Statistics
3) Embark on AI Interpretation Techniques
4) Present Results

7.2 Select Algorithms: Partial Dependency Plots (PDP) and SHAP

Given the available interpretation methods, two particular systems were chosen: PDP and SHAP methods. The PDP approach was chosen for its straightforward and simplistic representation. In the PDP, the partial dependency at a particular feature value is the average prediction if we assume all data points of that feature value (Friedman, 2001). In both correlated and uncorrelated cases, the plot presentations clearly show correlations between variables. Second SHAP interpretation is a suitable complement to the PD as it provides detailed (local) *feature* level information rather than global plot representations.

7.3 Display Descriptive Statistics

Once several thousand iterations of the simulation were run, the distribution of activities for each agent can be plotted on a frequency graph of the number of times an activity occurred. Although the results include data from both the herbivores and carnivores, for the sake of highlighting the interpretation methodology, this section focuses only on herbivore interpretation. First, activities that are available to each herbivore are defined:

Idle: 0
Move:1
Forage:2
Eat:3
Attack:4

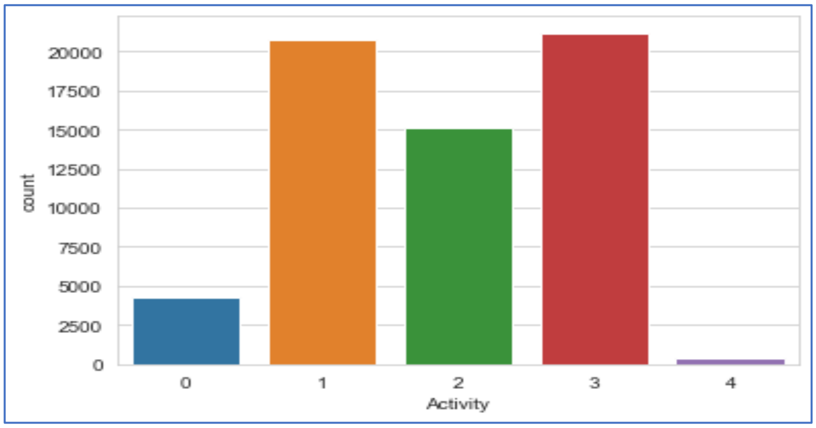


Figure 21: Distribution of Activities

Figure 21shows that of the five activities; *Move*, *Forage,* and *Eat,* have the highest frequency, and *Idle* and *Attack* have the least frequency. It is noted that at this time, there is no reference to the conditions that led to when the activities were applied, only that there is a particular frequency of behaviors. The next stage of the process predicts the next activity of the agent-based on prediction variables.

7.4 Embark on AI Interpretation Techniques

The following section presents work that was completed to achieve agent interpretability using Random Forest Classification and SHAP Interpretation. Code was developed in Python 3.7 and delivered using Jupyter Notebooks.

Table 22: PD and SHAP Code Link

Code Link	Description
https://github.com/paulsimvient/InterpretationCode	Section contains both random forest classification and SHAP value interpretation methods

7.4.1 Partial Dependence Plot Analysis

As discussed in section 7.2, the first AI interpretation method employed was the partial dependence plot (PDP). To plot the PDP, a random forest classifier is first trained using a cross-validation (CV) approach.CV is a technique used to test the effectiveness of a machine learning model; it is also a resampling procedure used to evaluate a model if there is limited data. In order to use the Randomized Search CV, model parameters are specified for searching the Python dictionary. Randomized Search CV implements to create a *Predict* and *Fit* method. Here in the CV model, parameters of the classifier are cross-validated to ensure accurate and optimized prediction. In the classification approach, the model predicts the probabilities of each class and determine which class has the highest probability for selection.

7.4.2 Generate Partial Dependency Plot Prediction Variables

Data were initially collected for each run of the model and stored within a CSV file. As each iteration of the simulation was run, data was collected by the time interval of a millisecond and gathered within the data file (see Table 23).

Table 23: Data Collected Within Simulation

Variable	Type	Description
identifier	int64	Unique Object ID
creature type	int64	Herbivore or Carnivore
Time	int64	Time in Seconds
reproduce	int64	Is it ready to reproduce?
CanGrow	int64	Variables set to grow
CanAttack	int64	Can attack opposing agent
Energy	float64	Level of Energy

MaxEnergy	int64	Maximum Energy
Size	float64	Size of agent
Age	float64	Age of Agent
Activity	int64	Activity Type
herbivores	int64	Number of Herbivores in Environment
carnivores	int64	Number of Carnivores in Environment
plants	int64	Number of Plants in Environment
herbivores_e	int64	Current Average Herbivore Energy
carnivores_e	int64	Current Average Carnivore Energy
plants_e	float64	Plant Energy (average)

First, several prediction variables are noted to be irrelevant. This is done through a process where we examine how variations, or perturbations, in these variables may influence activity change.

7.4.3 Partial Dependency Plot Data Preparation

The dataset preparation process divides data into training and testing data and begins by generating data based on the herbivore activity. Given the model is addressing only herbivore values, the variables related to carnivores (*carnivores*, *carnivores_e,* and *creature_Type*) are removed. Certain variables do not change throughout the model lifetime (e.g., *Identifier* and *MaxEnergy*) and can also be removed. *Time* and *Age* appear to be highly correlated, so either one of them can be selected as part of the training data (this is a machine learning rule for evaluating dependence).

7.5 Partial Dependency Plot (PDP) Analysis

After model training, a random forest model is applied for evaluating the most important variables; this is plotted in ascending order where the most important variable is plotted last. Given there are four activity classes, the PD is plotted for each class for analysis and observes how each variable affects activity.

7.5.1 *Activity 0: Idle*

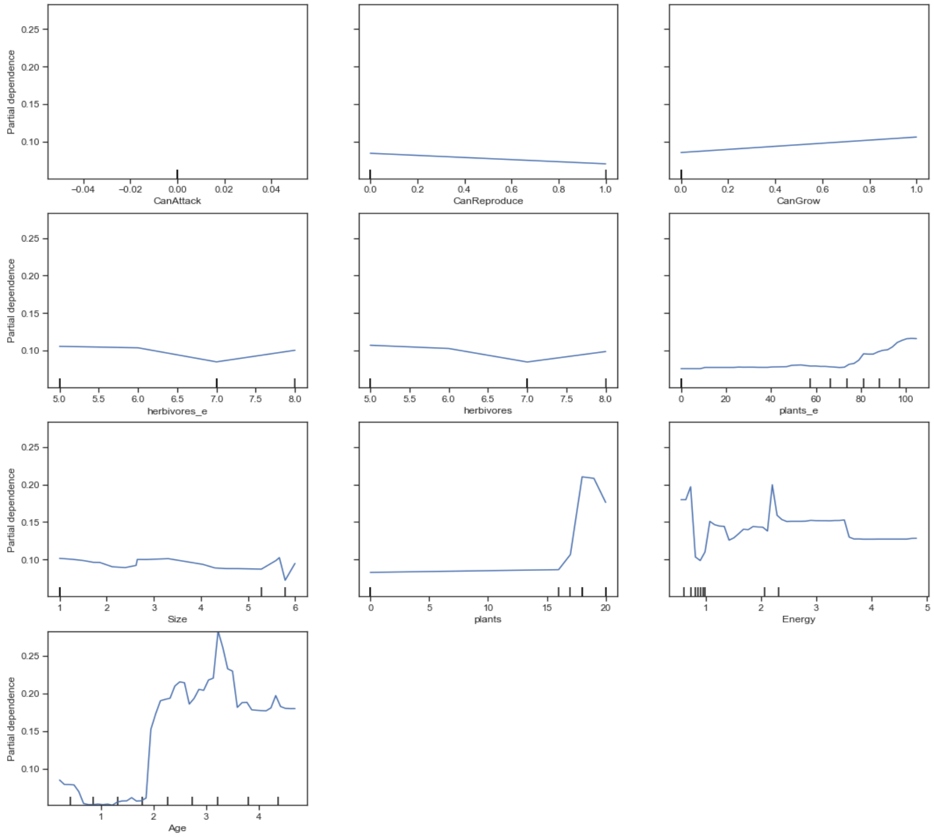


Figure 22: PDPs of Activity 0 (Idle) probability based on influencing variables

For the activity Idle (see Figure 22), the variable *Age* is the most important attribute. The PDP shows *Age* doesn't demonstrate a correlation until it reaches *Age*=2, and then shows a strong prediction correlation. For *Energy*, the probability did not show any trend, it decreases and increases without any trend, but after 2.5, the probability remains consistent. For *Number of Plants*, the probability is the same, but after 15, the *Number of Plants* probability exponentially increases.

7.5.2 Activity 1: Move

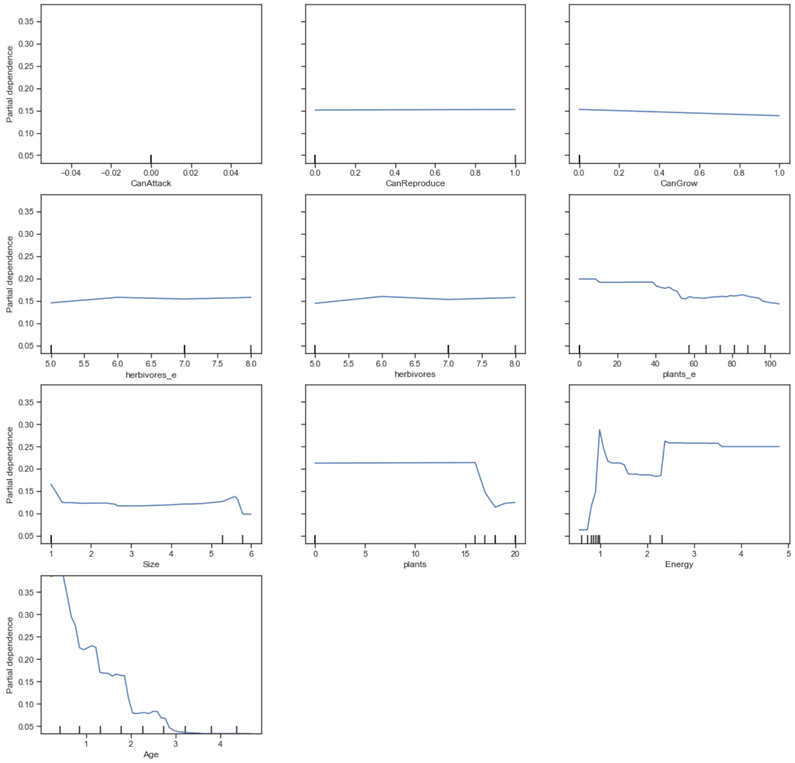


Figure 23: PDPs of Activity 1 (Move) probability based on influencing variables

For the activity *Move* (see Figure 23), the graph shows that as *Move* decreases, *Age* increases, i.e., it is negatively correlated with the *Move* activity. For *Energy*, its initial probability increases and then decreases, and after 2.5 it remains the same. For *Number of Plants*, the probability is consistent, but at roughly 15 plants, the probability for *Move* decreases. This might be because with the increase in plants, it is likely that moving isn’t necessary.

7.5.3 Activity 2: Forage

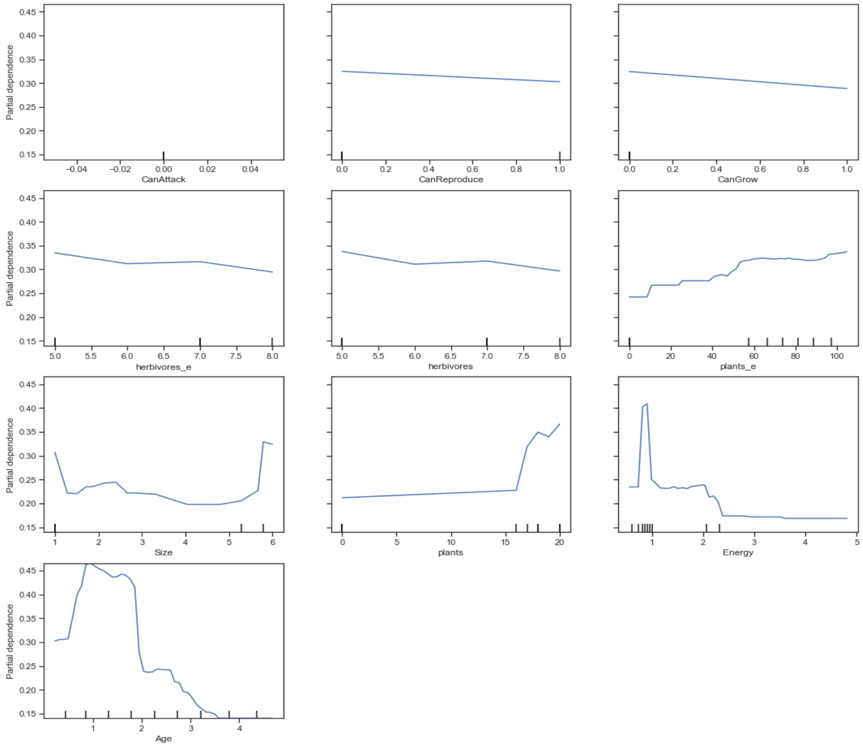


Figure 24: PDPs of Activity 2 (Forage) probability based on influencing variables

For the Activity Forage (see Figure 24), *Age*, the PDP shows that the *Forage* probability increases until value 1 and then decreases. For *Energy*, its dependence probability spikes early (1) then dissipates over time. For the *Number of Plants,* the probability is similar, but after 15 plants, the probability increases substantially.

7.5.4 *Activity 3: Eat*

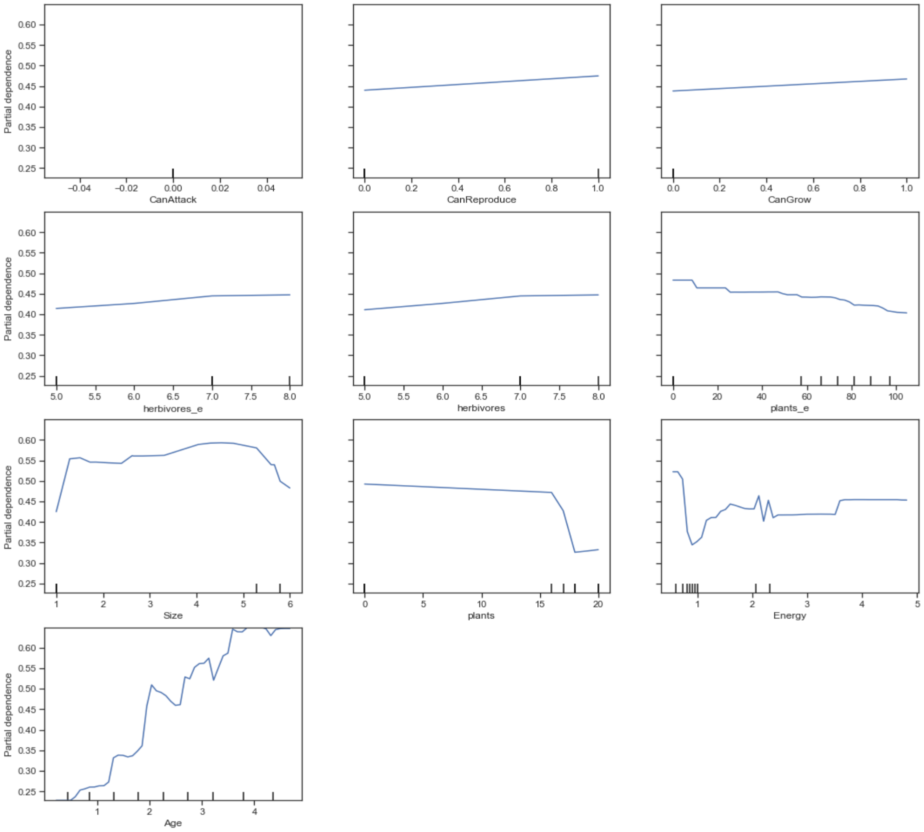


Figure 25: PDPs of Activity 3 (Eat) probability based on influencing variables

For the activity *Eating (see* Figure 25*)*, the PDP shows that the probability increases as the *Age* value rises and is highly positively correlated. For *Energy*, the probability decreases until $Age\ =\ 1$ and then increases as *Energy* value increases, but after $Age\ =\ 2.5$ the probability remains the same.

7.6 Partial Dependence Plot Results

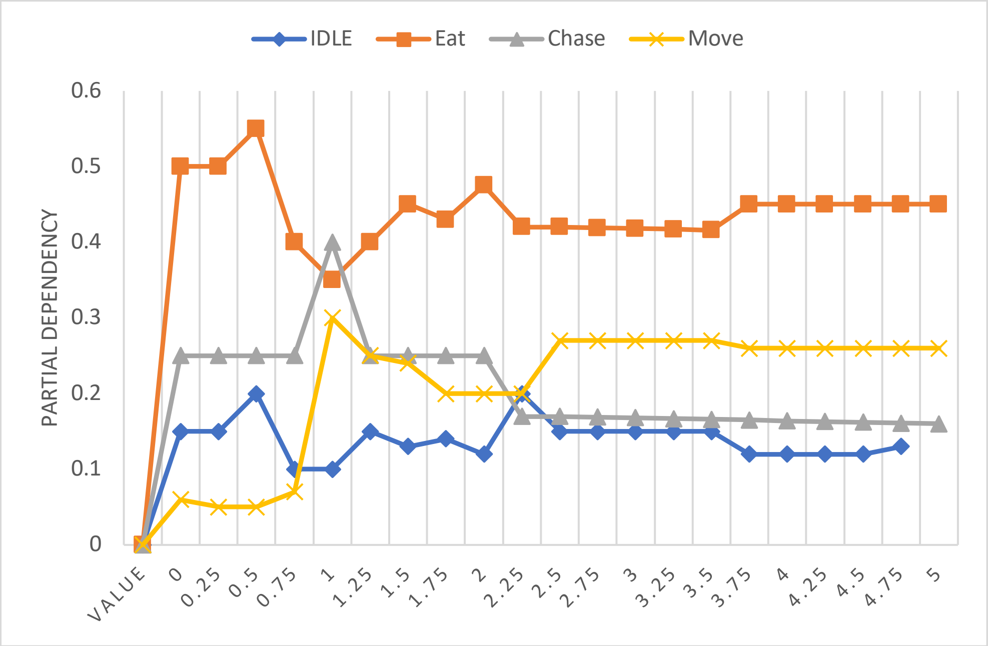


Figure 26: Energy (x-axis) as compared to Activity Partial Dependency (y-axis) to 4 Activities

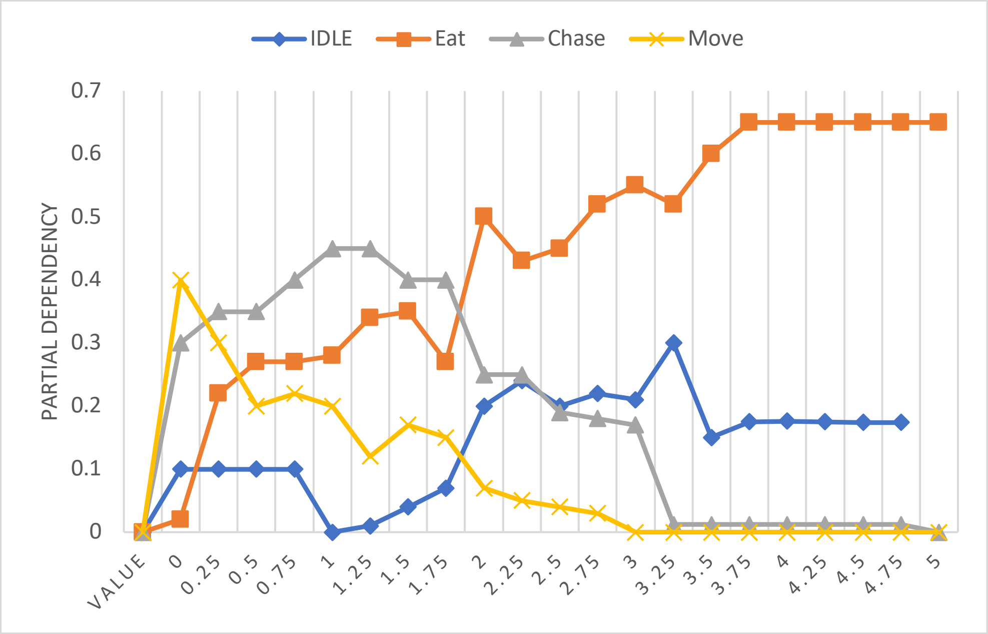


Figure 27 Age (x-axis) as compared to Activity Partial Dependency (y-axis) to 4 Activities

Upon analysis of the PDP, some interesting and more precise representations of how variables such as age and energy are influencing activities can be observed. For example, as *Age* increases, the agent has a high probability to *Eat*. Additionally, it does not *Forage* or *Move* as *Age* increases. On the other hand, *Age* does not appear to deter eating. In fact, as the *Age* increase, *Eat* increases even though the agent does not *Move*. Importantly, it must be ensured that behavior evaluation aggregates result from both *Age* and *Energy* when uncovering strategies. This is described in more detail in the analysis process.

7.7 SHAP Interpretation

SHAP (Shapley Additive explanations), looks at features within the model and assigns a prediction to each (Lundberg and Lee, 2017). Shapley (1953) values express the contribution that features have on the output of a model. There are two primary benefits of using SHAP values:

1. Global Interpretability: Describes the model from a global perspective across multiple features within an aggregated dataset.

2. Local Interpretability: Localizes a problem and describes the model in the local vicinity, rather than creating an explanation of the whole model.

7.7.1 Training the SHAP Model

A trained model for SHAP value calculation was built. SHAP algorithm has several methods for calculation of SHAP values for different models. Although several explainers are available (Tree, Gradient, Deep, Linear, Kernel), it was decided that a Tree Explainer would be used for several reasons.

1) sampling-based estimation variance is minimized, i.e., no need for a background dataset or select a subset of feature coalitions.
2) results are no longer skewed due to dependencies between features since these are contained in the tree structure (although under some circumstances, one would waive both of the previous benefits);
3) The run time is significantly faster. The Tree Explainer is used for tree ensemble models, given the approach is based on a random forest approach.

There are several choices for implementation, including XGBoost, LightGBM, CatBoost, scikit-learn Tree models (Random Forest, etc.), and pyspark tree models. LightGBM model is used as a low-performance impact approach to the calculation of SHAP values. LightGBM is a gradient boosting and tree-based learning model (Ke et al., 2017).

7.8 SHAP Global Interpretability (Variable Importance)

The concept of SHAP (see section 3.4.5) feature importance is fairly straight forward: Features with larger absolute Shapley values are more important than smaller ones. Given an interest in global relevance, Shapley absolute values are averaged per feature. Then, each feature is sorted by decreasing its relevance and plot the results (Lundberg, 2017). Then the SHAP values of every feature are plotted to demonstrate the most important model features. The Global Interpretability graph in Figure 28 is the sum of SHAP value magnitudes over all samples demonstrating variable importance.

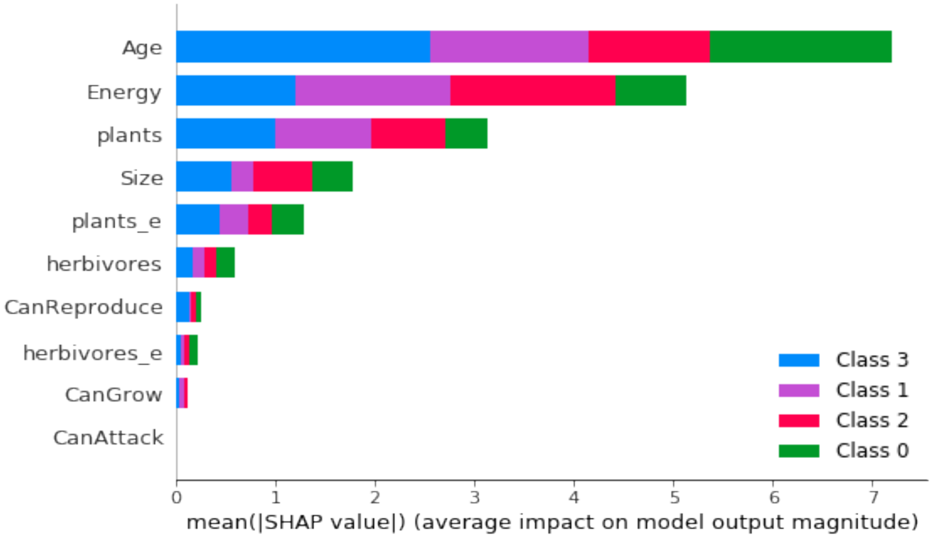


Figure 28: Global Interpretability Graph (Variable Importance)

The graph demonstrates that the most relevant feature is *Age* and the second one is *Energy*. Although the plot is interesting, it is a purely global outcome, i.e., it demonstrates the input variable effect as an aggregate to the total data set, and not specific observations. But it is worth noting that that given the *Partial Dependence Plot Results Age* is prioritized over *Energy* when considering the likelihood of activity.

7.9 SHAP Local Interpretability (Variable Importance)

The Summary Plot (Figure 29) is a density scatter plot showing features in SHAP values demonstrating the importance of each feature on the model output. The plot also shows the relationships of the predictors with the target variable (positive and negative). We have four activities and a plot for each activity and its analysis.

7.9.1 Explaining and Plotting Predictors and target Variables

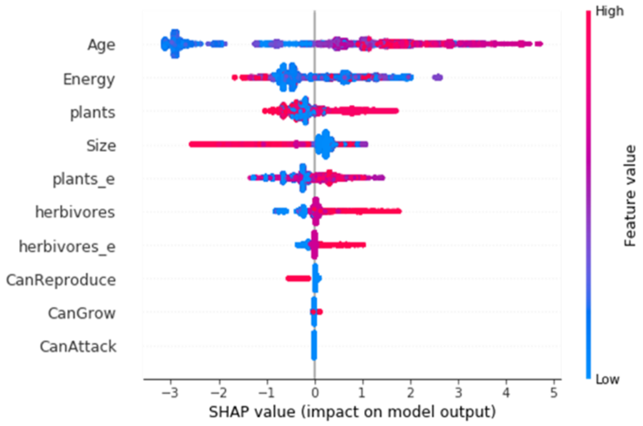


Figure 29: Example Summary Plot

This plot encompasses all points in the training data. It demonstrates the information, as discussed in Table 24.

Table 24: Reading the Summary Plot

Feature importance:	Ranked descending order values that describe the importance of the feature on model prediction
SHAP Value Impact:	The horizontal demarcation is showing how each value *is associated with a higher or lower prediction*, i.e., feature value over the entire dataset.
Color values	Color values where the red variable is higher predictions or Blue for lower predictions.
Feature Correlation	The vertical axis describes each feature as a high and *positive* correlation on the predictive quality rating. The red color implies "high" correlation, and Blue is "low" correlation.

From the diagram, it can be seen that *Age* and *Energy* have large effects on the prediction over the entire dataset (high SHAP value shown on the bottom axis). High *Age* values affect the prediction positively (red values on the right-hand side) while high Energy values affect the prediction negatively (red values on the left-hand side). For a more detailed explanation of Summary Plot results, please see the section in the APPENDIX: *Explaining SHAP Values as Individual Features.*

7.9.2 Activity 0: Idle Summary Plot

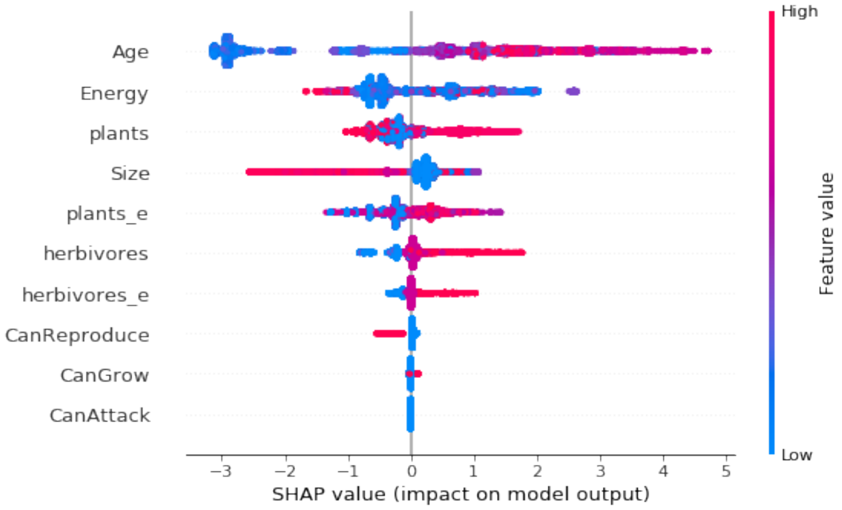


Figure 30: Activity 0 (idle) Summary Plot

For the *Idle* Activity, when *Age* value is high, then it is positively correlated with it, and its prediction accuracy increases. If the *Age* SHAP value is negative (blue color), then there is a negative correlation, and the SHAP values define a lower correlation. (This is same analysis as the random forest, where lower values have a lower probability, and after Age=2, the probability increases.) For *Energy*, if the value is high, then the probability is lower that it remains in an *Idle* activity. This is also the case for *Size,* where the agent size is negatively correlated with the *Idle* activity.

7.9.3 Activity 0 (Idle): Simplified plot

A simplified graph of the above figure is plotted. The Green color means a feature is positively correlated with the target variable and is negatively correlated when Red and no correlation is Blue.

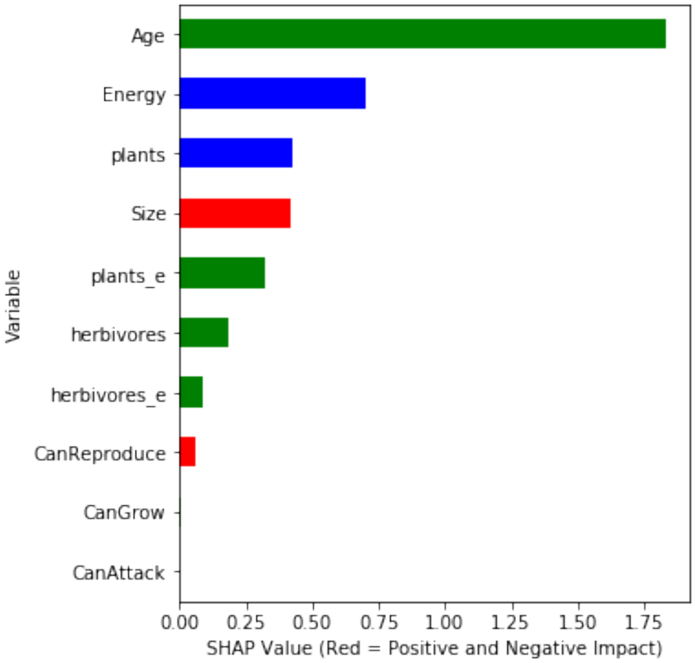


Figure 31: Activity 0 (idle) Simplified Plot

The next step is to interpret each variable correlation. *Age, plants_e, herbivores, and herbivores_e* are positively correlated, whereas *Size* and *CanReproduce* are negatively correlated to Activity 0.

7.9.4 *Activity 1 (Move) Summary Plot*

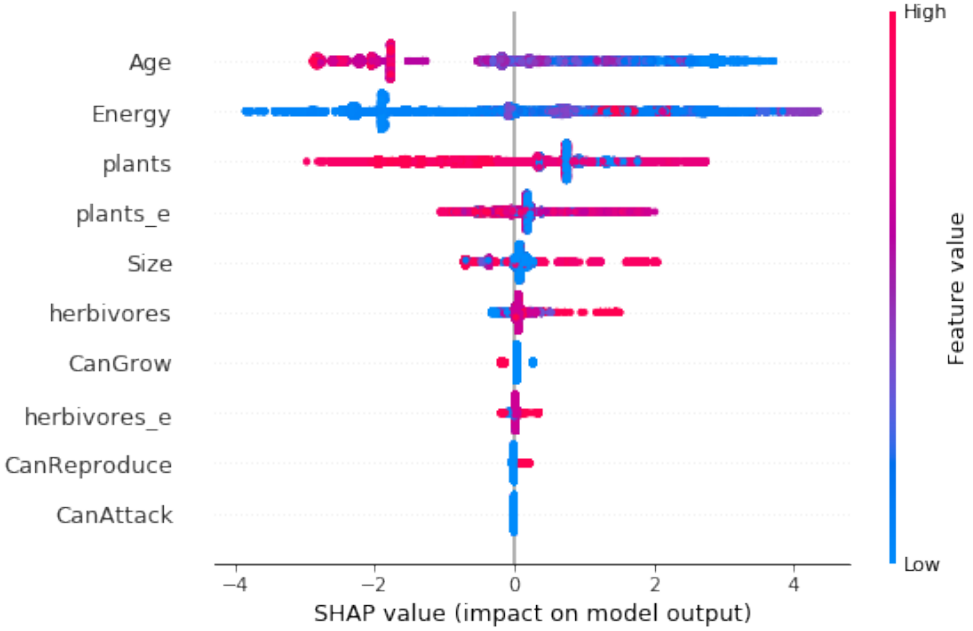


Figure 32: Activity 1(Move): Summary Plot

For Activity 1 (*Move*), it is noted that if *Age* is high, then the SHAP values are negative, indicating it is less likely to move as it gets older. This correlates to our Partial Dependency Plot results; whereas Age increases, less movement is expected (see Figure 32). One might also note that there is a distinct correlation between the number of plants and the desire to move where an increased number of plants is correlated with increased movement, and vice versa.

7.9.5 Activity 1 (Move): Simplified plot

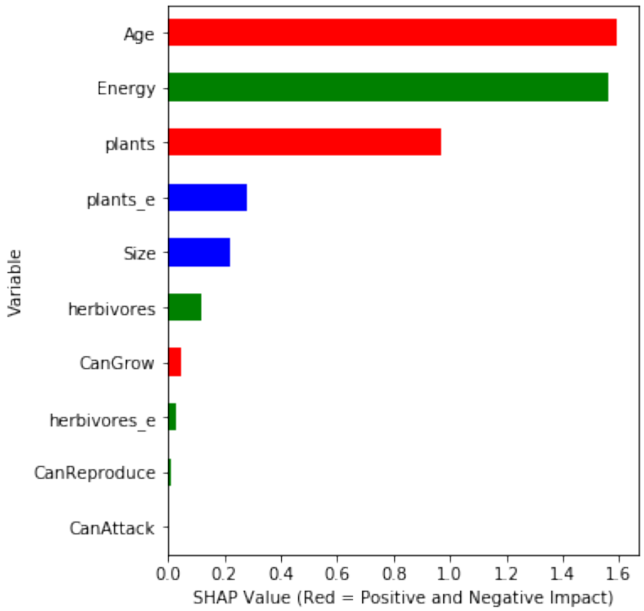


Figure 33: Activity 1 simplified plot

Figure 33 confirms Figure 32 results at a global scale. Here, *Age* and *number of plants* are negatively correlated, and *Energy* and the *number of herbivores* are positively correlated.

7.9.6 Activity 2 (Forage): Summary Plot

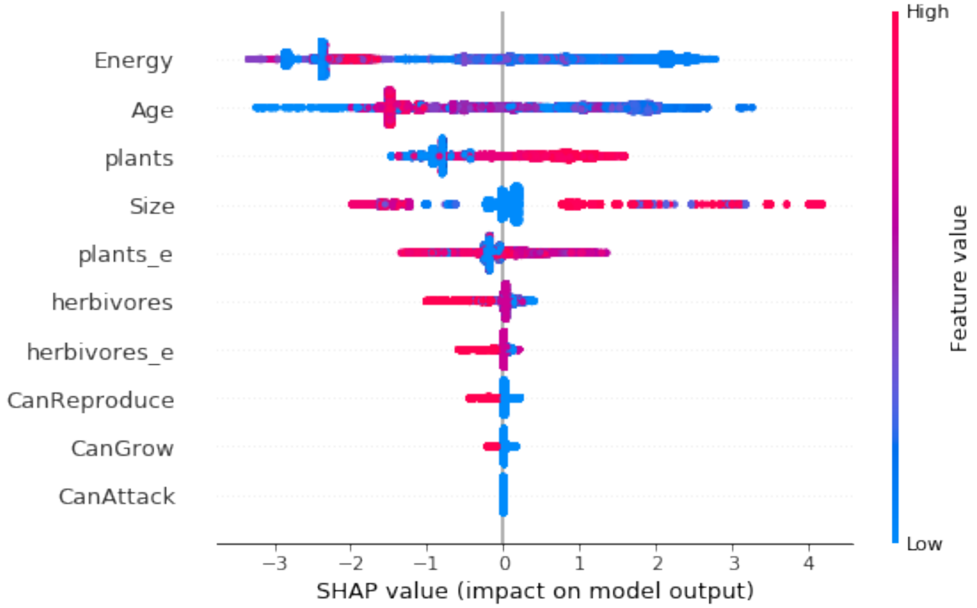


Figure 34: Activity 2 (Forage) Summary Plot

7.9.7 Activity 2 (Forage): Simplified plot

This plot describes the *Forage* activity stating that If *Energy* is high, then the probability for *Foraging* is low (blue color). This may occur because the *Energy* value has already been reached, and therefore there is less reason to *Forage*. It is also worth noting that as the *number of plants* increases, the likelihood of *Forage* also increases. One might also note that there is a high correlation between *Size* and *Forage* activity. What is of particular importance is that the graph demonstrates that the activity is not necessarily a predictor of the variable. So larger Size does not imply more *Forage* activity. It is more likely that more *Forage* behavior leads to a larger *Size*. And more plants also led to more *Forage* behavior.

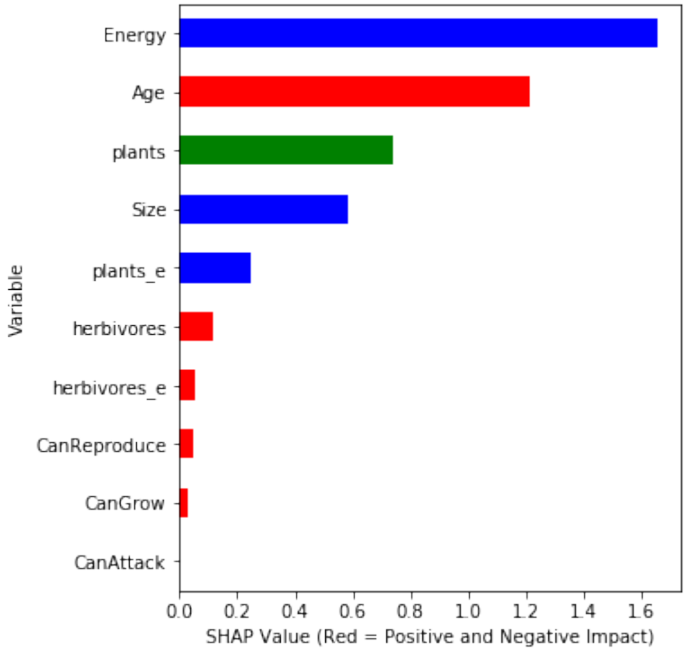


Figure 35: Activity 2 (Forage): Simplified Plot

For Activity 2 (Forage), Plants (number of plants) are positively correlated, and *Age, plants_e, herbivores, herbivores_e, CanGrow, and CanReproduce* are negatively correlated.

7.9.8 *Activity 3 (Eat): Summary Plot*

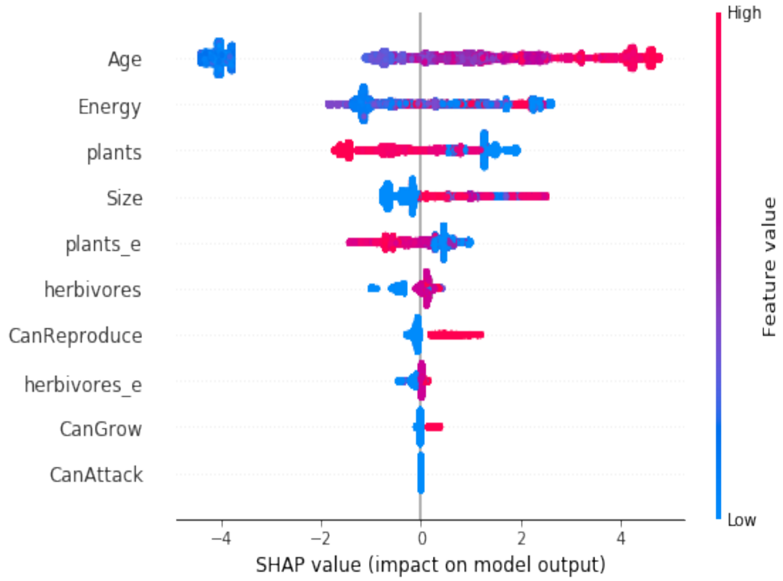


Figure 36: Activity 3 (Eat) Summary Plot

The plot demonstrates that if agent *Age* is high, then SHAP values are also high for *Eat* behavior, and they are positively correlated. Values are less clear for *Energy,* but for *Plants* (*number of plants*), it is negatively correlated. There are several interesting relationships that are represented here. First, *Age* should not necessarily be a reason for more eating, but there is another consideration, *Size*. If one were to consider what might be happening in the graph, imagine that as time progressed, more plants existed, and with that, the herbivore ate more, and *Size* increased when it *Ate* (noting the positive correlation). But as it ate, there were simply fewer plants (negative correlation). Also, given *Energy* was low, this would cause the herbivore to decide to *Eat* more (showing a negative correlation).

7.9.9 Activity 3 (Eat): Simplified Plot

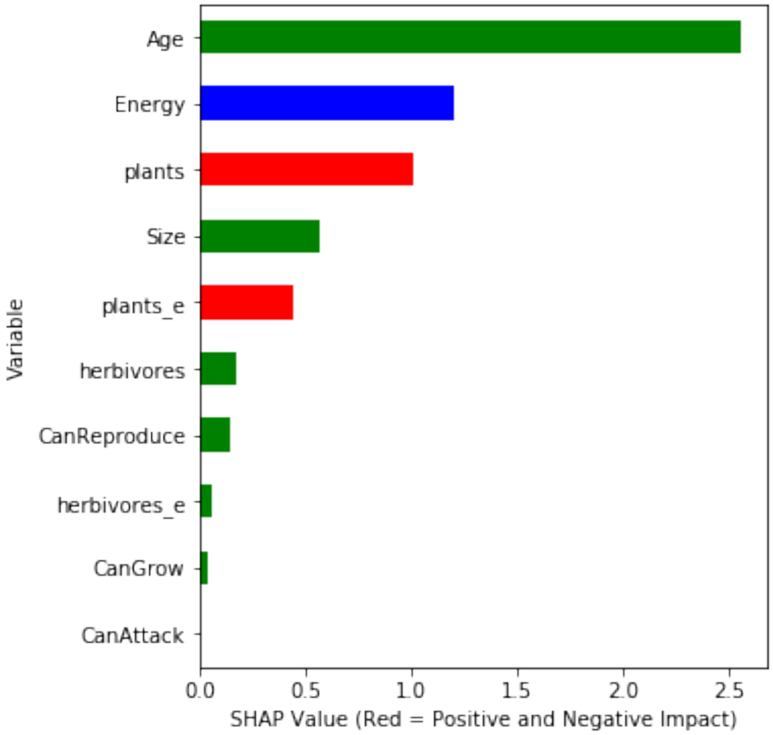


Figure 17: Activity 3 (Eat) Summary Plot

For Activity 3, *Age, Size, plants_e, herbivores, herbivores_e, Can Grow, and CanReproduce* are positively correlated, and *Plants* (number of plants) are negatively correlated.

7.10 Developing the State Representation: Finite State Machine

A FSM representation was built using results from the PDP and SHAP analysis. The finite state machine is a triple $M = (S, R, t)$, where S is a finite set of states., R is a finite set of symbols called the alphabet., $t: S \times A \rightarrow S$ is the transition function. The inputs to this function are the current state and the last input symbol (Arbib, 1969; Booth, 1967). While the function value $v(s, x)$ is the state, the automaton goes to from state s after reading the symbol x. The resultant FSM is then used as the basis for an agent model that is purely FSM based, which should mimic behaviors of its ML counterpart.

7.11 Developing a Conditional Map

The next phase was to construct the data into a format that can translate into a finite state representation. Probability trees are constructed that state the following: Given the current state of the agent (specifically its highest deterministic factors, *Age* and *Energy*), what is the likelihood it moves into one of the four states?

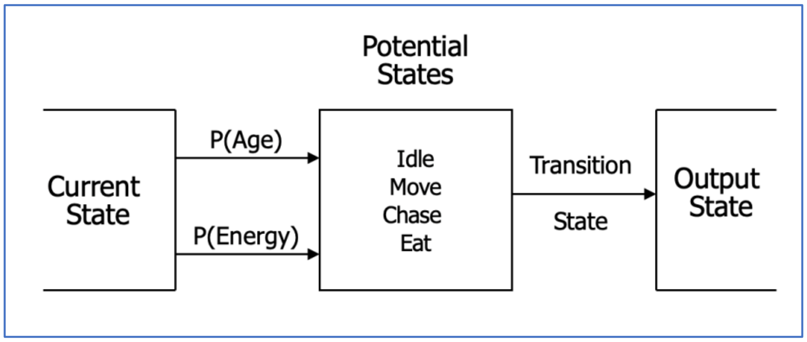


Figure 37: Conditional Map

Given this information is generated through the partial dependency map described in the previous section, the potential for selecting an activity based on the existing state of the agent is now more apparent. Noting earlier, the two primary predictors of its behavior were *Age* and *Energy*, so for simplicity's sake, these variables are highlighted as the most predictive characteristics of its potential state. Figure 38 and Figure 39are partial dependency plots of *Age* and *Energy,* respectively.

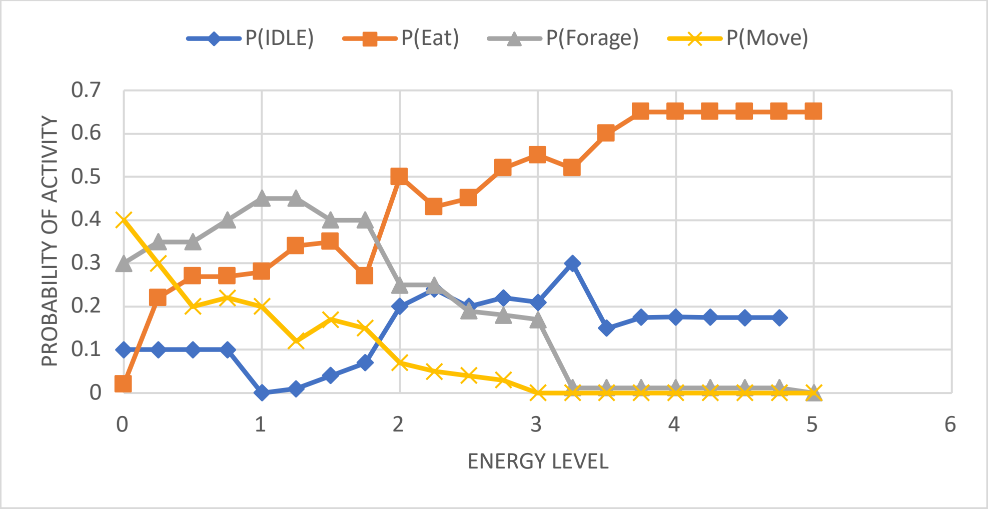


Figure 38: Energy Partial Dependency Plot

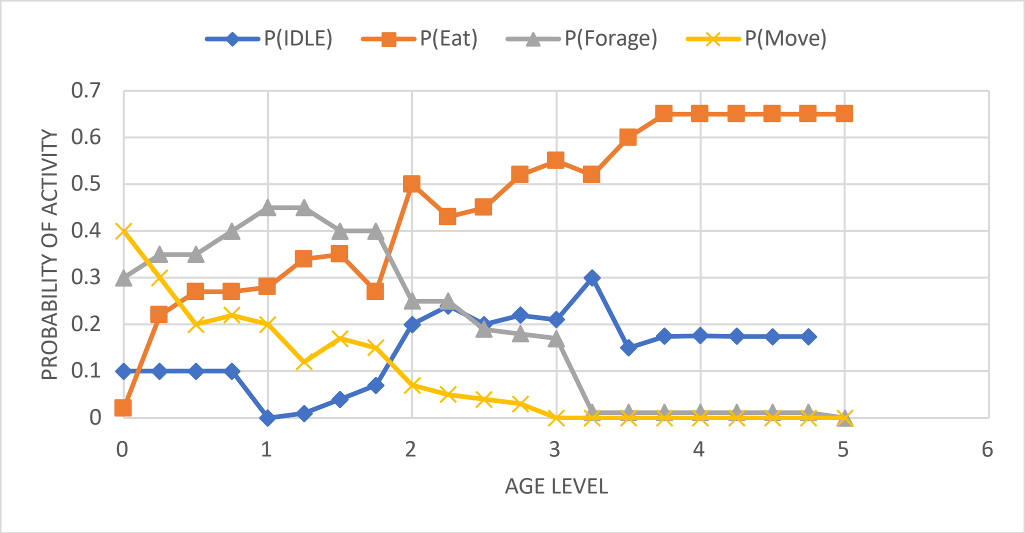


Figure 39: Age Partial Dependency Plot

The goal is now to create a distribution of states s for *Energy* and *Age* that lead to an action a. This allows for a range of parameters that can lead to a state. For example, when $Age\ =\ low\ (range\ [0-1])$ and $Energy\ =\ high\ (range\ [3-4])$, we can generate a set of probabilities the agent is participating in an action. These probabilities are listed in *Table 26: Conditional States Based on Interpretative AI Model*. The table provides state variable probabilities based on the *Age* and level of *Energy*. From these values, a set of states can be derived based on values. For example, based on Table 26, one can state the following: Given $Age\ (range)\ =\ X$ and Energy (range) = Y, the likelihood of selecting a state $S\ =((p)Idle \,|\, (p)Eat\ \mid\ (p)Chase\ \mid\ (p)Move\ )$.

Table 25: Ranges of Age and Energy

	Low	Low-Mid	High-Mid	High	Very High
Age	[0 - 1]	[1 - 2]	[2 - 3]	[3 - 4]	[4 - 5]
Energy	[0 - 1]	[1 - 2]	[2 - 3]	[3 - 4]	[4 - 5]

Table 26: Conditional States Based on Interpretative AI Model

AGE	ENERGY	IDLE	Eat	Forage	Move
Low	Low	14%	38%	34%	14%
Low	Low-Med	14%	35%	38%	14%
Low	High-Med	15%	37%	32%	15%
Low	High	15%	38%	32%	15%
Low	Very high	14%	40%	32%	14%
Low-Med	Low	10%	44%	37%	10%
Low-Med	Low-Med	9%	41%	41%	9%
Low-Med	High-Med	11%	43%	36%	11%
Low-Med	High	10%	44%	35%	10%
Low-Med	Very high	9%	46%	36%	9%
High-Med	Low	17%	45%	22%	17%
High-Med	Low-Med	17%	42%	24%	17%
High-Med	High-Med	18%	44%	20%	18%
High-Med	High	18%	45%	19%	18%
High-Med	Very high	17%	47%	19%	17%
High	Low	17%	51%	14%	17%
High	Low-Med	17%	49%	17%	17%
High	High-Med	18%	51%	12%	18%
High	High	18%	52%	11%	18%
High	Very high	17%	54%	11%	17%
Very high	Low	16%	56%	13%	16%
Very high	Low-Med	15%	54%	15%	15%
Very high	High-Med	17%	56%	10%	17%
Very high	High	17%	57%	9%	17%
Very high	Very high	16%	59%	9%	16%

Table 26creates a set of states and probabilities of the likelihood of being in one of the respective states. For example, if *Energy* is low and *Age* is low, we can expect a roughly 40% likelihood that the agent *Eats*, 35% it *Forages*, and 15% *Idle* and *Moves,* respectively. These conditions are the basis for the design of agent FSM and, thus, how we can describe its behavior in a readable narrative.

7.11.1 Translating Strategies into Cognitive Heuristics

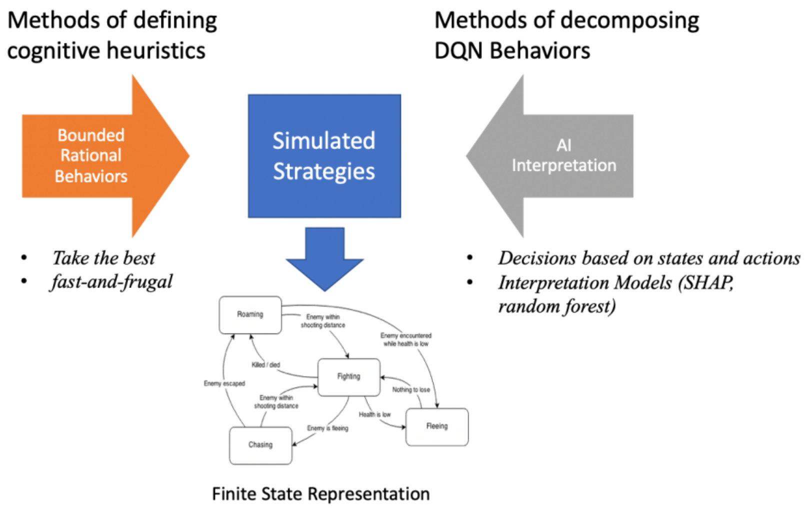

Methods of defining cognitive heuristics
Methods of decomposing DQN Behaviors
Bounded Rational Behaviors
Simulated Strategies
AI Interpretation
Take the best
fast-and-frugal
Decisions based on states and actions
Interpretation Models (SHAP, random forest)
Roaming
Enemy within shooting distance
Enemy encountered while health is low
Killed / died
Fighting
Enemy escaped
Nothing to lose
Enemy within shooting distance
Health is low
Fleeing
Enemy is fleeing
Chasing
Finite State Representation

Figure 40: Bridging Cognitive Heuristics and Explainable AI

Figure 40illustrates a basic representation of what is intended for review in this chapter. The goal is to derive boundedly rational behaviors (see 2.2.1) from an AI-Interpretation FSM. From the perspective of cognitive heuristics, some interesting potential strategies can be derived. For example, using the *take-the-best* heuristic, we can see that we can define behaviors from the perspective of "what is the highest probability action based on the existing state?" This is a single-reason decision rule, a type of heuristic where judgments are based on one "good" reason only, ignoring other indications (Gigerenzer & Gaissmaier, 2011). Using the take-the-best heuristic, one might deduce that, all things equal, an *Eat-Forage* strategy might be a viable approach to all conditions, where the agent fluctuates between the two based general conditions in the environment, where *fast-and-frugal* heuristics are useful in situations of uncertainty, while optimization is designed for risk-based situations. We may observe that under the duress of *Age*, even with high *Energy* expenditure, it may not be wise to *Forage*, although in this simple example, we can deduce that when *Age* increases, there may simply more things to eat within the environment. But high *Age* does not imply that agents take the opportunity to expend energy to find new food sources. In fact, one may say a strategy is simply "don't waste energy, eat what you have in front of you." This is further reinforced by the variable *Move,* which generally demonstrates movement is not advantageous, even in high *energy* states.

The premise itself is objectively clear. Interpretation techniques can derive both local and global level feature interpretation. And this provides some very compelling ways of creating behavior descriptions once the agents have created their respective optimal policies.

8 DISCUSSION AND FUTURE WORK

Although technical in nature, this work is no more an Artificial Intelligence dissertation than is Epstein and Axtell's work in generative societies is a computational system technology. The design, development, and analysis of this document is a methodology to generate computational strategies for social science research. Multi-agent systems, both homogenous and heterogenous, can benefit from developing LAISR agents. Once created, behaviors can be interpreted, analyzed, and represented as strategies for the social scientist to use as a method of modeling and measuring theory.

Starting with the first research question, "What methods can aid in the design, development, and analysis of hybrid ABM and reinforcement learning system in efforts to address challenges in ABM modeling?" In this section, I refer specifically to the portion of the agent that is using the deep reinforcement methods, the DRL-Agent. First, rational agents can exist in bounded conditions, i.e., states with limited understanding of the environment. These conditions have given rise to algorithms that have permitted us to draw relationships between Simon's concept of bounded rationality and the field of reinforcement learning; a DRL-Agent learns and makes decisions while not having a full understanding of its domain (Abel, 2019). Thus, it is argued that DRL can act as a method for modeling bounded rational agents. The Partially Observable MDP (POMDP) agents use a partial understanding of the environment to generate the true nature of the world state. The DRL-Agent, with its partially observable representation, must optimize actions based on the little it understands. Where rationality in human behavior is often limited by our existing knowledge, DRL-Agents are subject to restrictions on their understanding of the modeled environment (Abel, 2019). A key term that is used for this type of constraint is *computational rationality,* which assigns boundedness by resource constraints. Lewis et al.'s work (2014) describes a conceptual *Optimal Program Problem* (OPP) with three attributes: environment, a resource-constrained machine, and a utility function. My work has some overlap with the concept of computational rationality yet frames things somewhat differently; namely, DRL-Agents are not computationally resource-constrained; rather, they are limited by what their input sensors observe. Additionally, the difference between my model and that of Lewis' work is utility maximization. Lewis implied maximizing utility removes bounded rationality, which I offer is not the case. Utility maximization is a process for optimizing the decision process and is necessary for the model.

The argument for DRL-Agent and human cognitive processing is that both humans and agents do not have perfect rationality (Simon, 1955), and herein lies the fundamental importance of the approach – if full rationality does not exist in human behavior, nor should it in the representation of the behavior in silico. In the proposed model, Artificial Intelligence research provides some representation of learning and decision-making yet is still subject to realistic constraints on reasoning. The DRL-Agent empowers our ability to generate other forms of decision-making under realistic assumptions; these include the study of reinforcement learning (RL), a general problem construction in which agents must simultaneously learn about their environment while making good decisions in that environment. This type of boundedly rational, using reward signals rather than specific behaviors, can help social science imagine generalized questions without highly specific implementations.

8.1 Importance of this Work

Certainly, when one observes the level of detail it may take to generate DRL-Agents and resultant interpretations of their behavior, one may ask, is it worth it? After all, generating simple discrete agents with often complex and emergent results may be enough. Simon (1969) discusses AI in relation to decision-making where human cognition is limited in its ability to comprehend and respond to a vast amount of data; machines may be able to solve problems by simply running millions or even billions of steps to determine potential strategies. This is why this work, even its early implementation, may become highly valuable as a tool in the community. Here, we return to research question 2, "What AI-based research techniques can help to deconstruct behaviors of the proposed agent model into decision strategies that mimic attributes of human mental processing (i.e., 'fast-and-frugal)?" Interpretation modeling provides new and sometimes quite novel insights into behavior, and model interpretation methods such as SHAP and random forests can provide meaningful insights into the results of the DRL models. Succinctly, if we can allow the agent to run its course, its strategy development may provide insight into the way the brain creates its own behaviors. We may not be able to always create a perfect correlation between human decision-making and what the agent accomplishes, but that is not the point. The point is rather to allow the DRL-Agent to create its own strategies; sometimes, we may find a mapping to human decision-making, but sometimes we may also find new and interesting strategies that could enhance our understanding of some environment or state space. Rather than creating our representation of a boundedly rational agent representation, the agent itself makes that determination.

The heuristics and biases program made famous by Kahneman and Tversky (1972) examines rational choice and how it deviates from normal human behavior.

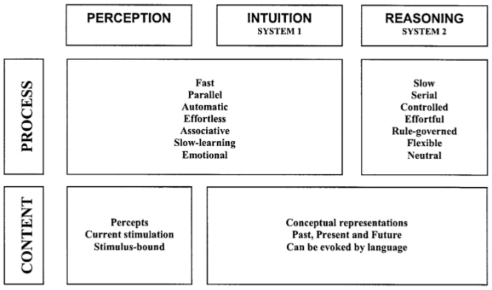


Figure 41: Kahneman Systems Revisited

For example, one may deduce that the DRL-Agent 's learning process is close to human intuition (See System 1: Figure 41), where it can assemble behaviors, create decisions with minimal effort, and generalize with limited cognitive processing. DRL-Agents train over hundreds of thousands, even millions of iterations, much like the brain does when selecting its own optimal approach to problem-solving. And like our own cognitive process, a DRL-Agent uses its knowledge to generalize over many situations. Kahnemanand Tversky (1972) state that human cognition relies on “some fast (in terms of time) and frugal (in terms of information acquisition and processing) heuristics” artificial agents in many ways, with limited knowledge, make their own ‘fast decisions.’ And with the model, we have tools that can aid in the decompositional behavior process.

These interpretation techniques, prevalent now in the AI community (i.e., SHAP and random forest interpretation models), provide glimpses into how social models in silico can be better understood. This is very important, where even precise problems are often very difficult to analyze (Fraenkel & Lichtenstein, 1981).

8.2 Revisiting Social Science Theory

When considering the relevance of the DRL-Agent model to social science theory, we must examine not only individual behaviors but that of groups of heterogeneous agents and how they cooperate, defect, or some middle strategy. Multiagent Reinforcement Learning algorithms (MARL) provide some ability to develop DRL-Agent behaviors but with new challenges in nonstationary policies (Papoudakis et al., 2019). Techniques to minimize the complexity of multi-agent learning using a "heuristic policy" (Bianchi, 2007) may help to advance the ability for multi-agent systems to learn together using DRL algorithms. For example, imagine an agent that is continuously refining its behavior by optimizing based on rewards, building new heuristics, then incorporating these heuristics into the new reward signal. This process could continuously test generalized theories by creating an almost infinite set of potential strategies, and with enough time, one could build a fully dynamic set of agents that respond to almost constantly changing conditions.

There are a few very key considerations in how these DRL methods may help us understand human mental processing. Das (2006) stated, there are no suitable definitions for intelligence, although understanding human divergences from decision norms might prove informative in the design of algorithms. Here, we can observe that heuristics and biases from Kahneman and Tversky's(2011) work can help us better understand how our cognitive processes may deviate from the models of rational choice. We could imagine that the heuristically biased processes in the brain are simply the brain's representation of a deep Q-neural network with the data it has available to it. Within the context of "ecological rationality," Mata et al. (2006) state that simple heuristic strategies can work well in many natural environments, suggesting that human rationality, much like an agent deep neural network, is an adaptive fit between its capacity to generate consistent and semi-optimal decisions across multiple environments. Todd et al. (2016) noted that in ecological rationality, top-down learning starts with a set of principles, and heuristics are built within uncertain environments and can be enhanced by refining complex models. This is, in fact, clearly similar to the way a refined neural network model accomplishes its goals. Gigerenzer & Goldstein (1996) also note heuristics are "ecologically rational" (capable of using existing information available within the environment) yet violate rationality norms. In fact, the two researchers have developed their own computational models that are fast, frugal, simple to operate even though they are computationally limited (Das, 2006).

But it is important to address the "how much is enough?" question as it relates to what a heuristic is in computational modeling. For example, DRL-Agent must straddle their domain knowledge where it is not so simple that it cannot produce a usable heuristic (under fitted) and one that is not generalizable enough to transfer knowledge into new domains (overfitted).   Horvitz (1987) notes the term *bounded optimality* seems to be the right goal, where an agent is bounded-optimal if its model is a solution given a constrained optimization problem presented by its architecture and the task environment.

8.2.1 Is the Heuristic a Neural Network Overfitting/Underfitting Problem?

This research leads us to how the human brain's non-rational behavior might be similar to what we see in a neural network. The brain's neural network 'under its' in order to generalize for as many conditions as possible. In fact, we have survived not simply by our adaptability but rather our generalizability. Our brains must learn to act quickly and efficiently with minimal resources, and so does the DRL-Agent. If its life – or how it is rewarded - depends on how it selects its next decision and future decisions, it must be careful not to simply be precise. It must also be prepared to adapt to a multitude of conditions and make choices that are optimal for its survival.

8.3 Future Work

Methods that are discussed in this dissertation provide some new and meaningful insights into the future of our understanding of complex behaviors in the context of social science theory. However, the work being presented has substantial room for growth. A few of the primary topics that are still underrepresented in this research are *causality, temporal detail,* and *heterogeneous decomposability*. First, the approaches that are being discussed related to model interpretation does not necessarily imply causality. Importantly, SHAP values do not identify causality. The relationships presented in SHAP and random forest graphs are important to demonstrate an association, but for the time being, that is all. However, the association is a very important first step towards uncovering causality. For example, observing general conditions where behaviors tend to occur can help to lay the foundation for deeper exploration of causality.

Second, when this work began, research related to time-based interpretation was still in its bourgeoning stages. Earlier (see section 3.4.2), an AI technique called *recurrent neural networks* (RNN) was referenced as a way to model and analyze time series and equivalent data. Currently, the DRL model considers time but only in a minimal context. For example, *Age* does not fluctuate; it only increases, so time-related to age is consistent and can be used to predict variables based on the SHAP model. *The energy,* on the other hand, fluctuates based on several other factors and cannot easily be associated with behaviors occurring over time. A more robust time-based interpretation technique may be important in future work.

Third, behavior interpretation is still fairly coarse; specifically, it can be difficult to evaluate a set of strategies for agents who are of the same type yet act differently. Even if all the agents share a common neural network, this can still be a challenging DRL problem to overcome, given the converge of behaviors is unlikely. Luckily, local representations such as SHAP Summary Plot helps to identify large behavior distributions by showing how easy it is to predict features. The feature values identify how much impact each feature has on the model output and show the positive and negative relationships of the predictors with the target variable (see Figure 29). For example, we can use SHAP data to determine how relevant a particular action is to the model (*feature value*), and in doing so, we can generate a distribution of possible actions that fall within the SHAP feature distribution. We can also use the SHAP data to generate a prioritized list of actions that influence a feature, and therefore can weigh the potential for an action to take place. But there is still much work to be done here.

8.4 Applications of the Actor-Interpreter Model

It is believed that the notion of allowing agents to develop behaviors and to observe conceptually how they can help us uncover strategies, behaviors, and insights entice our community for years to come. AI agents may exist to help us understand that we are no longer bound by our own cognition. Within the social science discipline, we can expand our ability to design social scenarios and observe how new and diverse behaviors emerge as DRL models can uncover novel ways to refine their understanding of environments. These agents can evaluate the domain space, then provide suggestions towards problem-solving. This is a uniquely important aspect of social science, where through traditional computational approaches, our agents are often pre-wired with discrete rules. The approach can enhance the research that spans modeling, real-time simulation, and training, as well as learning strategies using intelligent tutoring and remediation. Several new fields are emerging where the use of an Actor-Interpreter model can be of great use, where there may be rules and robust solutions for an agent (Vu et al., 2019).

8.4.1 Modeling Agents in Gaming and Entertainment Environments

Game technologies have begun to enhance mechanics through the use of reinforcement learning agents. Creating semi-intelligent non-playable game characters through RL has supported game scalability and complexity (Juliani, 2018). ML-Agents, which is used in this dissertation, is an open-source platform providing tools for policy and off-policy learning. One would likely notice that the same dynamics generated in gaming environments can also be applied to other forms of entertainment, including VR experiences and movie special effects. Game testing can be enhanced through the design of DRL-Agents/Interpreter and examined through the proposed interpretation process.

8.4.2 Wargaming and Decision Support

Wargames are analytic games that simulate aspects of warfare at the tactical, operational, or strategic level. They are used to examine warfighting concepts, train and educate commanders and analysts, explore scenarios, and assess how to force planning and posture choices to affect campaign outcomes. Several examples have been developed and can execute various types of wargames, including scenario exercises, tabletop map exercises, games, and computer-supported exercises. For example, RAND (Linick et al., 2020) developed Hegemony, a wargame designed to teach U.S. defense professionals how different strategies could affect key planning factors in the trade space at the intersection of force development, force management, force posture, and force employment. The use of DRL-Agent/Interpretation offers ways of examining new strategies for wargaming and decision support professionals.

8.4.3 Department of Defense (DoD) Training Systems

The DoD is the development of modernized simulation technologies for training, which includes live, virtual, and constructive training exercises. The approach to the development of these complex exercises includes the demands of adaptive AI is constantly changing scenarios that exist in ambiguity and chaos. For example, the DoD's Synthetic Training Environment (STE) provides training as a service through immersive simulation and gaming technologies where AI (and reinforcement learning) can optimize human performance and support operational and training scenarios (Hubal, 2017). Generating dynamic training scenarios and examining DRL-Agent/Interpreter solutions can transform the way that training is accomplished for the next generation warfighter.

8.4.4 Research: Inverse Generative Social Science (IGSS)

Similar to the DRL-Agent approach, IGSS is not to build entire agent types, but rather, to encode rules, parameters, and possible mathematical principles, one would search the space for the most appropriate agent architectures. Vu and Epstein (2019) develop an IGSS approach using Genetic Programming, Decision Trees, Causal State Modeling, and Machine Learning and AI. Certainly, a bridge exists, which may lead to similar definitions, methods, and theoretical underpinnings. The use of the DRL-Agent and Interpreter can be an important addition to the IGSS community.

This dissertation finishes with a short fictional narrative about a socially evolving agent named Darla. Although quite fantastical, it provides a guiding light to what may be possible when we think about the future of AI and behavioral attributes in the coming decades. Agents may themselves be part of either fictitious or non-fictitious narrative and must build their own decision processes to adapt and thrive in their surroundings. The future is certainly exciting and left with great opportunities to explore.

Darla, as a young biomorph, was no larger than the size of a house cat. In a quiet corner of her playpen, she touched the follicles of her abrasive hair against everything within her reach. Her single lidless eye stared intently at each object she handled, sometimes for several minutes at a time, in what appeared to be a gentle meditative trance. Her experimentation was much like how a baby explores its world. It was mesmerizing to watch. To view a biomorph interacts with its surroundings was like watching a work of art paint itself. No two biomorphs were alike, often not even remotely alike. Some were large string-like beings that wrapped around their observations like a snake eating its prey. Others were shapeless beings that moved through the environment like purposeful gelatin. Still, other biomorphs resembled the form and function of multi-legged creatures, akin to small insects.

Darla's behaviors were often difficult to fathom. Unlike the gates, walks, crawls, and undulations we are accustomed to seeing in earth creatures, biomorphs were highly experimental in shape and mobility. It was not unusual to see a creature use two or three large protrusions scrape against the ground while smaller foot-like projections would push from behind. Sometimes a biomorph could move in circles or poke at the ground for hours on end, demonstrating no logical reason for its behavior. Other times, after days of seemingly random movements, an entity would suddenly stop and begin an exquisite orchestration of movement. She was now roughly 4.5 billion trillion episodes old, or more specifically, 123 super-epics. Unlike typical sentient life, it did not make much sense to measure age by cycles around the sun. The biomorph had a distinct advantage as compared to earth organisms – it did not need to live a lifetime to grow and evolve. Where, for example, it can take 100,000 years before a mutation in the human gene pool to become an adaptive trait, biomorphs were able to learn, evolve, and re-design themselves in a picosecond.

The skin that surrounded Darla had a soft and wool-like attribution. Although it appeared to be fibrous in texture, her skin was actually a dense array of micro-cilia that that were in constant motion, often too small for the eye to observe unless under a microscope. However, the skin was highly advanced, allowing it to work in conjunction with other cilia to create what appeared to be synthetic appendages. It took a few days for her body to reach a stable state. She would observe and experiment with its attributes, often holding on to it for several days, then would begin a new transformation. At the moment, she appeared happy with cilia tentacles growing from the top of her head. Her current experimentation did not include a need for mobility, and hence legs and arms were not well-developed. All the while, the world seemed to be a beautiful place for Darla to explore.

It was difficult to say whether her experience was one of joy, inquisitiveness, or simple observation, but Darla never tired from the endless exploration of her world.

APPENDIX

Introduction: Discussion of Coding Process

DRL-Agent Code Links

The first two code bases are in reference to the homogeneous and heterogeneous reinforcement learning systems.

Reference Section	Code Link
https://github.com/paulsimvient/Homogeneous-MultiAgent	Homogeneous Multi-Agent Reinforcement Learning
https://github.com/paulsimvient/Sheep-Wolf	Heterogeneous Multi-Agent Reinforcement Learning

Interpretation Models

The following section presents work that was completed to achieve agent interpretability using Random Forest Classification and SHAP Interpretation. Code was developed in Python 3.7 and delivered using Jupyter Notebooks.

Reference Section	Code Link	Description
Interpreting the Heterogenous LAISR Model	https://github.com/paulsimvient/InterpretationCode	Section contains both random forest classification and SHAP value interpretation methods

Explaining SHAP Values as Individual Features

For the reader, it may be somewhat difficult to decipher the SHAP graph. Therefore, some examples are provided to attempt to explain predicted Activity with SHAP Values. First, the activity *Move* as a target is selected. Predictions are explained with the SHAP values, and a Summary Plot are developed for one target pointer activity.

Activity 0 (Idle):

According to the model probability of this activity is 0.0012, which is low. The graph is a decomposition of a single point in the Summary Plot.

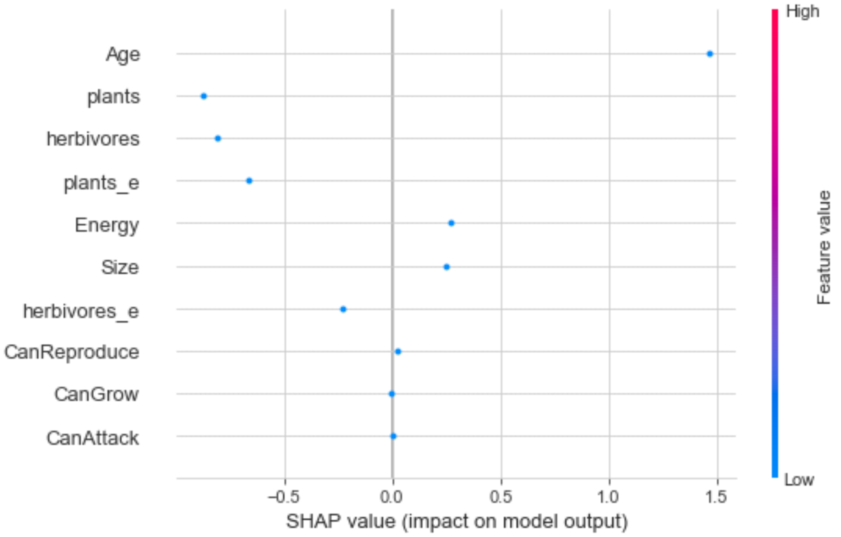


Figure 42: Idle Activity Individual Feature

Age has the highest positive SHAP value 1.5, which is increases probability. *Energy* and *Size* also have positive values, but values are low, so they are not increasing the probability much higher. *Number of Plants*, herbivores, and *Plant Energy* (plants_e) have higher negative SHAP values, which are decreasing probability.

Activity One (Move):

According to the model probability of the *Move* activity is 0.9942which is very high, and also, the target activity is one. The summary plot is as follows

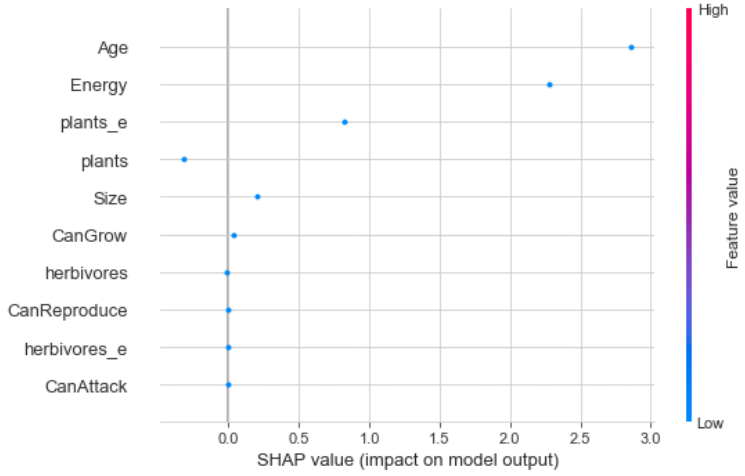


Figure 43: Move Activity Individual Feature

Age has 3.0 shape value, and *Energy* has 2.3 value, which pushes probability very high, and only plants have a lower negative value. So that is why it has a high probability for this activity.

Activity 2 (Forage):

According to the model, the probability of this activity is 0.004.

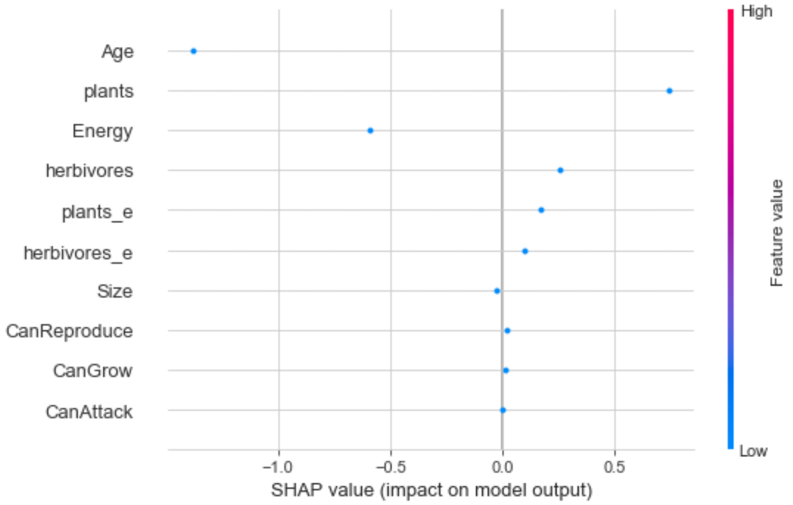


Figure 44: ForageActivity Individual Feature

Age has a very high negative SHAP value, which is decreasing the probability, and energy is also decreasing the probability. *Number of Plants* (plants), *Number of Herbivores* (herbivores), and *Herbivore Energy* (herbivores_e) is increasing probability.

Activity 3 (Eat):

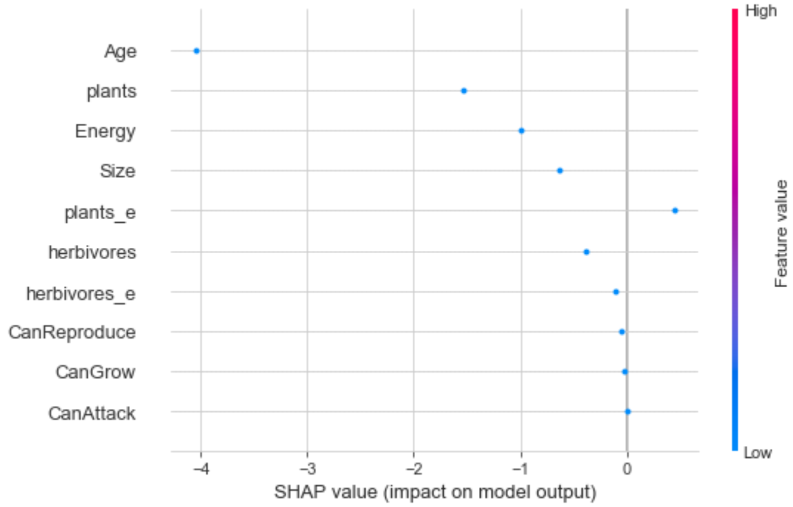


Figure 45: Eat Activity Individual Feature

All the prediction variables are decreasing probability, and only plants_e is increasing probability. But the overall impact is negative, so the probability of this *Activity* is zero. Overall, the *Age* factor is the most important feature in the prediction, age is positively correlated with eating activity, and middle-aged stays idle at most among all ages. *Energy* is the most non-trending feature, which sits at generally the same position, whereas in *Forage* activity with high energy, the probability is lowest.

Examining Feature Importance

After the initial model training, certain important features can be identified according to the Light Gradient Boosting Machine.

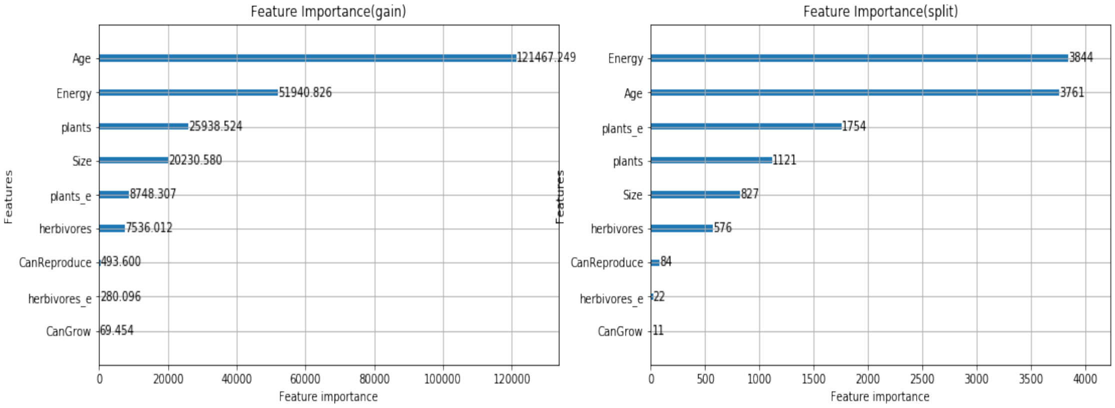


Figure 46: Light Gradient Boosting Machine Results (Gain vs. Split)

Figure 46 displays two graphs: gain and split. The *gain* is the contribution of the equivalent feature in the model computed by each feature's influence in each tree model. The *split* is the number of times the feature appears within the model, i.e., the number of times each feature was used when creating the trees. Very succinctly, one may notice that the gain and split models are not equivalent. Given that, accuracy is determined when generating shap values

REFERENCES

Abbott R., Hadžikadić M. (2017). Complex Adaptive Systems, Systems Thinking, and Agent-Based Modeling. In: Hadžikadić M., Avdaković S. (eds) Advanced Technologies, Systems, and Applications. Lecture Notes in Networks and Systems, vol 3. Springer, 1-8.

Abel, D. (2019). Concepts in Bounded Rationality: Perspectives from Reinforcement Learning. (Master's Thesis).

Agogino A., Tumer K. (2004). Efficient Evaluation Functions for Multi-Rover Systems. In the Genetic and Evolutionary Computation Conference, Seatle, WA. 1-12.

Alahi, A., Goel, K., Ramanathan, V. Robicquet, A., Fei-Fei,L., Savarese, S. (2016). Social LSTM: Human Trajectory Prediction in Crowded Spaces. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 961-971.

Alexander, P. A., Schallert, D. L., & Hare, V. C. (1991). Coming to terms: How researchers in learning and literacy talk about knowledge. Review of Educational Research, 61(3), 315-343.

Ali, S., and Shah, M. (2010). Human action recognition in videos using kinematic features and multiple instance learning., IEEE Trans. Pattern Anal. Mach. Intell., 32(2). 288–303.

Allard C.R (2003). Effective Decision-Making in the High-Tech Service Innovation Process, Doctoral Dissertation, Maastricht University, Maastricht, Datawyse/Maastricht University Press.

Almeida, A., Azkune, G. (2018). Predicting Human Behaviour with Recurrent Neural Networks. Appl. Sci. 2018, 8, 305.

Anderson, L. W., & Krathwohl, D. R. (2001). A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives: Complete Edition. New York: Longman.

Andrews, R. Diederich J. and Tickle, A. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems, Vol. 8, no. 6. 373-389

Andrews, R., Diederich, J., Tickle, A. (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. In Knowledge Based Systems.

Andrews, R., Diederich, J., Tickle, A. (1995). Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks, Knowledge-based Systems, Vol. 8, No. 6, 373- 389.

Antonosvsky, A. (1993). Complexity, Conflict, Chaos, Coherence, Coercion and Civility Social Science and Medicine, 969-974.

Apeldoorn, D., Kern-Isberner, G. (2017). A Discrete-Heuristic Learning Approach for Finding and Exploiting Heuristics in Unknown Environments, Proceedings of the Thirteenth International Symposium on Commonsense Reasoning, {COMMONSENSE} 2017, London, UK, November 6-8.

Arbib, Michael A. (1969). Theories of Abstract Automata (1st ed.). Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Ardema, M. Heymann, and N. Rajan (1985)., Combat Games, Journal of Optimization Theory and Applications, Vol. 46, No. 4, 391–398.

Ardema, M., Rajan, N. (1987). An approach to three-dimensional aircraft pursuit-evasion. Computers & Mathematics with Applications, Vol. 13, no. 1-3. 97–110.

Baddeley, B. (2008). Reinforcement learning in continuous time and space: Interference and Not Ill Conditioning is the main problem. Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics, Vol. 38, Nov 4, August 2008, 950-956.

Beattie, C., Leibo, J., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Petersen, S. (2016). DeepMind Lab.

Bastani, O., Pu, Y., Solar-Lezama, A. (2018), Verifiable Reinforcement Learning via PolicyExtraction, NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2499–2509.

Been, K., Khanna, R., Koyejo, O. (2016). Examples Are Not Enough, Learn to Criticize! Criticism for Interpretability. Advances in Neural Information Processing Systems, Curran Associates, Inc. Nips2016, 2280-2288.

Bellemare, M. G., Dabney, W., &Munos, R. (2017). A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707. 06887.

Bellemare, M. G., Dabney, W., &Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. Arxiv Preprint Arxiv:1707.06887.

Bellman, R. E. (1957). Dynamic Programming. Princeton Press.

Bernoulli, B. (1738). Exposition of a New Theory on the Measurement of Risk, Econometrica, 22(1): 23–36. Doi:10.2307/1909829.

Berretty, P. M., Todd, P. M., &Martignon, L. (1999). Categorization by Elimination: Using Few Cues to Choose. In G. Gigerenzer, P. M. Todd, & the ABC Research Group (Eds.), Simple Heuristics That Make Us Smart (235–254). New York: Ox-Ford University Press.

Bertsekas, Dimitri P. (1995). Dynamic Programming and Optimal Control. Vol. 1 and 2. Athena Scientific.

Bhavnani R, Miodownik D. (2009). Ethnic Polarization, Ethnic Salience, and Civil War. Journal of Conflict Resolution.; 53(1):30–49.

Bianchi, R., Ribeiro, C., Costa, A. (2007). Heuristic Selection of Actions in Multiagent Reinforcement Learning. IJCAI International Joint Conference on Artificial Intelligence. 690-695.

Bideau, B., Kulpa, R. Vignais, N., Brault, S., Multon, F., Craig, C. (2010). Using Virtual Reality to Analyze Sports Performance. Computer Graphics and Applications, IEEE 30(2): 14-21.

Bonabeau, E. (2002). Agent-based modeling: Methods and Techniques for Simulating Human Systems. Proceedings of the National Academy of Sciences, 99 (Supp 3):7280–7287.

Booth, Taylor L. (1967). Sequential Machines and Automata Theory (1st ed.). New York: John Wiley and Sons, Inc. Library of Congress Card Catalog Number 67-25924.

Brandstätter, E., Gigerenzer, G., &Hertwig, R. (2006). The Priority Heuristic: Making Choices without Trade-offs. Psychological Review, 113, 409–432.

Brandstätter, E., Gigerenzer, G., &Hertwig, R. (2008). Risky Choice with Heuristics: Reply to Birnbaum (2008). Johnson, Schulte-Mecklenbeck, and Willemsen (2008). and Rieger and Wang (2008). Psychological Review, 115(1), 281–289.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1993). Classification and Regression Trees. New York: Chapman & Hall.

Brenner, T. and Werker, C. (2006). A Practical Guide to Inference in Simulation Models. Papers on Economics and Evolution 2006-02, Philipps University Marburg, Department of Geography.

Briggs, R. A. (2019). Normative Theories of Rational Choice: Expected Utility", The Stanford Encyclopedia of Philosophy (Fall 2019 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/fall2019/entries/rationality-normative-utility/>.

Broadbent, B. E. (1958). Perception and Communication. New York, Ny: Pergamon Press. Doi: 10.1037/10037-000.

Brocas I., Carrillo J.D. (2010). Neuroeconomic Theory, Using Neuroscience to Understand Bounds of Rationality, Voxeu.org, March.http://www.voxeu.org.

Bruch E., Mare R. (2006). Neighborhood Choice and Neighborhood Change. American Journal of Sociology;112(3):667–709.

Brunswik, E. (1943). Organismic Achievement and Environmental Probability. Psychological Review, 50, 255-272.

Burger, A., Oz, T., Crooks, A., & Kennedy, W. G. (2017). Generation of Realistic Megacity Populations and Social Networks for Discrete-Heuristic Modeling. Proceedings of the 2017 International Conference of The Computational Operational Planning Society of the Americas on - CSS 2017, 1–7.

Busoniu, L., Babuska, R. and Schutter, B. D. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning, Trans. Sys. Man Cyber Part C, Vol. 38. 156–172.

Carbonell, J. (1983). Learning by Analogy, in Machine Learning: An Artificial Intelligence Approach, Michalski, R., Carbonell, J., and Mitchell, T. (eds.), San Francisco: Morgan Kaufmann.

Castaneda, A. (2016). Deep Reinforcement Learning Variants of Multi-Agent Learning Algorithms. Master's Dissertation, School of Informatics, University of Edinburgh, 2016.

Castelvecchi, D. (2016). Can We Open the Black Box of AI? Nature.com, https://www.nature.com/news/can-we-open-the-black-box-of-ai-1.20731

Caudell, T. P., and Mizell, D. W. (1992). Augmented Reality: An Application of Heads-up Display Technology to Manual Manufacturing Processes. In Proceedings of the Twenty-Fifth Hawaii International Conference on Systems Science, Kauai, Hawaii, 7th-10th Jan. 1992, Vol. 2. 659-669.

Champion, E., Bishop, I., and Dave, B. (2011). The Palenque Project: Evaluating Interaction in an Online Virtual Archaeology Site. Virtual reality: 1-19.

Chan, S. (2011). Complex Adaptive Systems. Website: https://tinyurl. com/yaccpw93.

Chang, M. (2011). Discrete Heuristic Modeling and Computational Experiments in Industrial Organization: Growing Firms and Industries "in silico." Eastern

Economic Journal, 37(1), 28-34. Retrieved from http://www. jstor.org/stable/41239491.

Chang, S. (2006). The Systematic Design of Instruction, Educational Technology Research and Development 54(4):417-420, DOI: 10. 1007/s11423-006-9606-0.

Chimeh, M.K., Richmond, R.(2018). Simulating Heterogeneous Behaviours in Complex Systems on GPUs, Simulation Modelling Practice and Theory, Volume 83, 3-17.

Christopher, M. B. (2006). Pattern Recognition and Machine Learning. Springer.

Cioffi-Revilla, Claudio (2014). Introduction to Computational Social Science: Principles and Applications (Texts in Computer Science). Springer London. Kindle Edition.

Clancey, W. J. (1986). From Guidon to Neomycin and Heracles in Twenty Short Lessons, AI Magazine. 7, Number 3, 40-60.

Claus, C., Boutilier, C. (1998). the Dynamics of Reinforcement Learning in Cooperative Multi-Agent Systems. in 15th National Conference on Artificial Intelligence, 746–752.

Clemente, a. V., MartıNez, H., Nicolas C., and Chandra, a. (2017). Efficient Parallel Methods for Deep Reinforcement Learning. Corr, Abs/1705. 0486.

Cramer, H. S. M. (2004). Usability Evaluation and Context Analysis to Support Development of Virtual Reality Systems. Master Dissertation. Faculty of Science, University of Amsterdam, Netherlands.

Craven, M. (1996). Extracting Comprehensible Models From Trained Neural Networks, Ph. D. Dissertation, Department of Computer Sciences, University of Wisconsin-Madison.

Craven, M. W. (1996). Extracting Comprehensible Models From Trained Neural Networks, Ph.D. Dissertation, Department of Computer Sciences, University of Wisconsin-Madison.Crites, R. H.,Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In: D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.) Advances in Neural Information Processing Systems 8. 1017–1023. MIT Press.

Crooks, a., Malleson, N., Manley, E., Heppenstall, a. (2019). Agent-Based Modeling and Geographical Information Systems, Spatial Analytics and GIS, Sage Publications. 126.

Cruz-Neira, C., Sandin, D. J. and Defanti T. a. (1993). Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the Cave. in Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (Siggraph '93): 135-142.

Cummings, P. and Crooks, A.T. (2020). Development of a Hybrid Machine Learning Agent Based Model for Optimization and Interpretability, in Thomson, R., Bisgin, H., Dancy, C., Hyder, a. and Hussain, M. (Eds), 2020 International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation, Washington DC., 151-160.

Das, S. (2006). On Agent-Based Modeling of Complex Systems: Learning and Bounded Rationality, Elsevier Preprint.

Davis, P., Kulick, J., Egner, M. (2005). Implications of Modern Decision Science for Military Decision-Support Systems, Rand Corporation. Ed. 1.

Dawid, H., Gemkow, S., Harting, P., Van Der Hoog, S., &Neugart, M. (2012). the Eurace@Unibi Model: A Discrete-Heuristic Macroeconomic Model for Economic Policy Analysis, 12-13.

Dawid, H., and Neugart, M. (2011). Discrete-Heuristic Models for Economic Policy Design. Eastern Economic Journal, 37.

Dejong, G., and Mooney, R. (1986). Explanation-Based Learning: An Alternative View, Machine Learning, 1:145-176, Reprinted in Shavlik, J. and Dietterich, T., Readings in Machine Learning, San Francisco: Morgan Kaufmann, 1990, 452-467.

de Jong, T., & Ferguson-Hessler, M. G. M. (1996). Types and Qualities of Knowledge. Educational Psychologist, 31(2). 105-113.

Deangelis D., Mooij W. (2005). Individual-Based Modeling of Ecological and Evolutionary Processes. Annual Review of Ecology, Evolution, and Systematics. 36:147–168.Delage, E., and Mannor, S. (2010). Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. Operations Research, 58(1):203–213.

Dhami, M. K. (2003). Psychological Models of Professional Decision-Making. Psychological Science, 14, 175–180.

Dick, W. (1996). the Dick and Carey Model: Will It Survive the Decade? Educational Technology Research and Development 1).44, Number 3, 1996 Issn 1042-1629.

Dietterich, T. G. (1999). Hierarchical Reinforcement Learning with the Maxq Value Function Decomposition. J. Artif. Intell. Res., 13, 227-303.

Diuk, C., Schapiro, a., Córdova, N., Ribas-Fernandes, J., Niv, Y., &Botvinick, M. (2013). Divide and Conquer: Hierarchical Reinforcement Learning and Task Decomposition in Humans. New York, Ny, Us: Springer-Verlag Publishing, 271-291.

Dorado, J., Rabunãl, J., Santos, a., Pazos a., and Rivero, D. (2002). Automatic Recurrent and Feed-Forward Ann Rule and Expression Extraction with Genetic Programming, Proceedings 7th International Conference on Parallel Problem Solving From Nature, Granada.

Eberlen, J., Scholz, G., &Gagliolo, M. (2017). Simulate This! an Introduction to Discrete-Heuristic Models and Their Power to Improve Your Research Practice. International Review of Social Psychology, 30(1), 149–160.

Egidi, M., & Marengo, L. (2004). Near-Decomposability, Organization, and Evolution: Some Notes on Herbert Simon's Contribution. in M. Augier & J. G. March (Eds.). Models of a Man: Essays in Memory of Herbert a. Simon (335-350). Cambridge, Ma, Us: MIT Press.

Elshawi, R., Al-Mallah, M., &Sakr, S. (2019). On the Interpretability of Machine Learning-Based Model for Predicting Hypertension. BMC Medical Informatics and Decision-Making, 19.

Emer, J., Gloy, N. (1997). a Language for Describing Predictors and Its Application to Automatic Synthesis. in 24th Annual International Symposium on Computer Architecture.

Epstein, J. (2006). Generative Social Science: Studies in Agent-Based Computational Modeling (Princeton Studies in Complexity) (5-6). Princeton University Press.

Epstein, J., Cummings, D., Chakravarty, S., Singa, R., Burke, D. (2004). Toward a Containment Strategy for Smallpox Bioterror: An Individual-Based Computational Approach. Generative Social Science: Studies in Agent-Based Computational Modeling. 1-55.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., Kavukcuoglu, K. (2018). ImpalA: Scalable Distributed Deep-Rl With Importance Weighted Actor Learner Architectures. Arxiv E-Prints. European Conference on Machine Learning.

Fagiolo, G. and Roventini, a. (2007). Macroeconomic Policy in DSGE and Agent-Based Models. Revue de L'Ofce, 124:67–116.

Fehérvári, I., &Elmenreich, W. (2010). Evolving Neural Network Controllers for a Team of Self-Organizing Robots. J. Robotics, 2010, 841286:1-841286:10.

Forster, M. (1999). How Do Simple Rules `Fit to Reality' in a Complex World?. Minds and Machines. 9. 543-564. 10.1023/A:1008304819398.

Francisco S. Melo. (2007). Convergence of Q-Learning: A Simple Proof, Proceedings of the European Control Conference Kos, Greece, July 2-5.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M., and Pineau, J. (2018). An Introduction to Deep Reinforcement Learning, Foundations and Trends in Machine Learning: Vol. 11, 3-4.

Fraenkel, a., Lichtenstein, D. (1981). Computing a Perfect Strategy for N*N Chess Requires Time Exponential in N. ICALP. Friedman, M. (1953). Essays in Positive Economics, Chicago Press.

Friedman, J. H. (2001). Greedy function approximation: A Gradient Boosting Machine. Annals of Statistics: 1189-1232.

Funkhouser, T. Jot, J. -M. and Tsingos, N. (2004). Survey of Methods for Modeling Sound Propagation in Interactive Virtual Environment Systems. Presence - Teleoperators and Virtual Environments.

Galán, J. M., Izquierdo, Izquierdo, S. S., Santos, J., del Olmo, R., López-Paredes, A., Edmonds, B. (2009). Errors and Artefacts in Agent-Based Modelling. Journal of Artificial Societies and Social Simulation. 12 (1): 1.

Gatto, M., Maue, J., Mihalák, M., &Widmayer, P. (2009). Shunting for Dummies: An Introductory Algorithmic Survey. Robust and Online Large-Scale Optimization.

Gelenbe, E., Hussain, K., Kaptan, V. (2005). Simulating Autonomous Agents in Augmented Reality Journal of Systems and Software Archive. 74 Issue 3, 255-268.

Gigerenzer, G., Todd, M.P., ABC Research Group (Eds.). (1999). Simple Heuristics That Make Us Smart. New York: Oxford University Press.

Gigerenzer, G. & Goldstein,D. (1996). Reasoning the Fast-and-frugal Way: Models of Bounded Rationality. Psychological Review. 62. 650-669.

Gigerenzer, G., &Kurzenhäuser, S. (2005). Fast-and-frugal Heuristics in Medical Decision-making. In R. Bibace, J. D. Laird, K. L. Noller, & J. Valsiner (Eds.), Science and medicine in dialogue: Thinking through particulars and universals (p. 3–15). Praeger Publishers/Greenwood Publishing Group.

Gigerenzer G. (2016). Towards a Rational Theory of Heuristics. In: Frantz R., Marsh L. (Eds) Minds, Models and Milieux. Archival Insights into the Evolution of Economics. Palgrave Macmillan, London.

Gordon, G., Ahissar, E. (2011). Hierarchical Curiosity Loops and Active Sensing, 2012 Special Issue, Neural Networks. 32, August 2012, 119-129.

Gorr, W. L. (1994). Editorial: Research Prospective on Neural Network Forecasting. International Journal of Forecasting, 10(1):1–4

Granovetter, Mark. (1978). Threshold Models of Collective Behavior, American Journal of Sociology 83 (May): 489-515.

Grazzini, J., Gatti, D. (2013). Paper on The Development of Mabm Mark Ii: The Input-output Network in The Crisis Macro Agent-based Model. Crisis Project Deliverable D3. 3, Universit Cattolica Del Sacro Cuore, Milano.

Grazzini, J., Richiardi, M. G. (2013). Consistent Estimation of Agent-based Models By Simulated Minimum Distance. Laboratorio R. Revelli Working Papers Series 130, Laboratorio R. Revelli, Centre For Employment Studies. Greek, R., Hansen, L. A. (2013). Questions Regarding the Predictive Value of One Evolved Complex Adaptive System for a Second: Exemplified by the SOD1 Mouse. Progress in Biophysics and Molecular Biology, 113, 231-253.

Grazzini, J., Richiardi, M. G., and Tsionas, M. (2017). Bayesian Estimation of Agent-Based Models. Journal of Economic Dynamics and Control, 77:26 – 47.

Green, L., Mehr, D. R. (1997). What Alters Physicians' Decisions to Admit to the Coronary Care Unit?. Journal of Family Practice, 45(3), 219–226.

Grey, J. (2002). Human-computer Interaction in Life Drawing, a Fine Artist's Perspective. Proceedings 6th International Conference on Information Visualisation: 761-770.

Guestrin, C., Michail G. Parr, R. (2002). Coordinated Reinforcement Learning: Proceedings of the Nineteenth International Conference on Machine Learning. Icml '02. San Francisco, Ca, USA: Morgan Kaufmann Publishers Inc., 227–234.

Gunning, D. (2017). Explainable Artificial Intelligence (Xai), Defense Advanced Research Projects Agency, Darpa/i20.

Ha, D., Schmidhuber, J. (2018). World Models, Arxiv Preprint Arxiv:1803. 10122, 4-5.

Hayles, K. (2008). Electronic Literature: New Horizons for the Literary, Notre Dame: University of Notre Dame Press, 45-50.

Heckbert S, Baynes T, Reeson A. (2010). Agent-based Modeling in Ecological Economics. Annals of the New York Academy of Sciences. 1185(1):39–53.

Hengst B. (2011). Hierarchical Reinforcement Learning. In: Sammut C., Webb G. We. (Eds) Encyclopedia of Machine Learning. Springer, Boston, Ma, Abstract.

Henrya, J. A. G. And Polysb, N. F. (2010). The Effects of Immersion and Navigation on the Acquisition of Spatial Knowledge of Abstract Data Networks. International Conference on Computational Science (ICCS '2010): 1737-1746.

Hernandez-Leal, P., Kaisers, M., Baarslag, T., Munoz De Cote. E. (2017). A Survey of Learning in Multiagent Environments: Dealing with Non-stationarity.

Hertwig, R., Davis, J. N., &Sulloway, F. J. (2002). Parental Investment: How an Equity Motive Can Produce Inequality. Psychological Bulletin, 128, 728–745.

Hertwig, R., Hoffrage, U., &Martignon, L. (1999). Quick Estimation: Letting the Environment Do the Work. In G. Gigerenzer, P. M. Todd, & the ABC Research Group, Evolution and Cognition. Simple Heuristics That Make Us Smart (P. 209–234). Oxford University Press.

Hidasi, B., Karatzoglou, A., Baltrunas, L., &Tikk, D. (2016). Session-based Recommendations with Recurrent Neural Networks. CoRR, abs/1511.06939.

Hinton, G. E., and Sejnowski, T. J. (1999). Unsupervised Learning: Foundations of Neural Computation In: G. E. Hinton and T. J. Sejnowski (Eds.) Unsupervised learning: Foundation computation, MIT Press, Cambridge, MA, vii-xv.

Hobbs, J. R. (1985). Granularity. In: Proc. of IJCAI, Los Angeles, USA, 432–435.

Holland J. (1992). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. Cambridge, Ma: Mit Press.

Howard, Richard Author (1960). Dynamic Programming and Markov Decision Processes. MIT Press.

Horvitz, E. J. (1988). Reasoning about Beliefs and Actions under Computational Resource Constraints. In T. Levitt, J. Lemmmer, & L. Kanal (Eds.), Uncertainty in Artificial Intelligence 3. Amsterdam: North Holland.

Hu, J. and Wellman, M. P. (1998). Multiagent Reinforcement Learning: Theoretical framework and an algorithm. Proceedings of the Fifteenth International Conference on Machine Learning. IEEE Computer Society. 285–285.

Hubal, R.& Parsons, T. (2017). Synthetic Environments for Skills Training and Practice.

Hutchinson JM1, Gigerenzer G. (2005). Simple Heuristics and Rules of Thumb: Where Psychologists and Behavioral Biologists Might Meet. Behavioral Processes. May 31; 69(2):97-124.

Hutter, M. (2004.) Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability. Berlin: Springer.

Ijsselsteijn, W. A., Ridder, H. D., Freeman, J. and Avons, S. E. (2000). Presence: concept, determinants, and measurement. Proceedings of SPIE human vision and electronic imaging: 520-529.

Jaderberg, M., Czarnecki, WM., Dunning, I., Marris, L., Lever, G. (2018). Human-level Performance in First-person Multiplayer Games with Population-based Deep Reinforcement Learning. Submitted, 2018.

Janssen, C., Gray, W. (2012). When, What, and How Much to Reward in Reinforcement Learning-Based Models of Cognition. Cogn Sci. 2012 Mar;36(2):333-58.

Johnson, M., Hofmann, K., Hutton, T., &Bignell, D. (2016). The Malmo Platform for Artificial Intelligence Experimentation, in Ijcai. 4246-4247.

Johnson, W. L. (1994). Agents that Learn to Explain Themselves. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 1257-1263.

Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M. and Lange, D. (2018). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809. 02627.

Kaelbling, L.P., Littman, M. L., Cassandra, A.R (1998). Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence, 101(1):99–134.

Kahneman D., Tversky A. (1972). Subjective Probability: A Judgment of Representativeness. In: Staël Von Holstein CA.S. (eds) The Concept of Probability in Psychological Experiments. Theory and Decision Library (An International Series in the Philosophy and Methodology of the Social and Behavioral Sciences), vol 8. Springer, Dordrecht.

Kahneman, D. (2003). Maps of Bounded Rationality: Psychology for Behavioral Economics. American Economic Review, 93 (5).

Kahneman, D. (1973). Attention and Effort. Englewood Cliffs, Nj: Prentice-hall Inc.

Kahneman, D. (2003). Maps of Bounded Rationality: Psychology for Behavioral Economics. American Economic Review, 93 (5): 1449-1475.

Kahneman, D. (2011). Thinking Fast and Slow, Deep Learning, and Aihttps://lexfridman.com/daniel-kahneman/.

Kamruzzaman S., Islam, M. M. (2006). An Algorithm to Extract Rules from Artificial Neural Networks for Medical Diagnosis Problems, International Journal of Information Technology, Vol. 12, No. 8. 41−59.

Kamruzzaman, S. M., Hasan, a. (2010). Rule Extraction Using Artificial Neural Networks.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Y, Q., Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree, Advances in Neural Information Processing Systems 30 (Nips 2017), Pp. 3149-3157.

Kempka, M., Et Al. (2016). VizDoom: a Doom-based AI Research Platform for Visual Reinforcement Learning. In Computational Intelligence and Games (Cig), 2016 Ieee Conference on (1-8).

Kennedy, W. (2012). Modelling Human Behavior in Agent-based Models. Chapter 9 of Current Geographical Theories for Agent-based Modelling, Edited by Michael Batty, Alison Heppenstall, and Andrew Crooks. Springer.

Kim, J. Canny, J. (2017). Interpretable Learning for Self-driving Cars by Visualizing Causal Attention, the IEEE International Conference on Computer Vision (ICCV), 2017. 2942-2950.g

Kitani, K. M. Ziebart, B. D. Bagnell, J., Herbert. M. (2012). Activity Forecasting. In Computer Vision–ECCV 2012, Springer., 201–214.

Kok, J., Vlassis, N. (2004). Sparse Tabular Multi-Agent Q-learning. In Annual Machine Learning Conference of Belgium and the Netherlands, 65–71. Kok, J., Vlassis, N. (2006). Collaborative multi-agent reinforcement learning by payoff propagation. Journal of Machine Learning Research, 7: 1789–1828.

Kolodner, J. (1993). Case-Based Reasoning, San Francisco: Morgan Kaufmann,

Korteling Je, Brouwer Am, Toet a. A Neural Network Framework for Cognitive Bias. Front Psychol. 2018; 9:1561. Published 2018 Sep 3. Doi:10.3389/fpsyg.2018.01561

Koutnık, J. Schmidhuber, J. and Gomez, F. (2014). Evolving Deep Unsupervised Convolutional Networks for Vision-Based Reinforcement Learning. In Proceedings of the 2014 conference on Genetic and evolutionary computation, 541–548. ACM, 2014.

Kubler, S., Voisin, A., Derigent, W., Thomas, A., Rondeau, E., Främling, K.(2014). Group Fuzzy AHP Approach to Embed Relevant Data on Communicating Material. Comput. Ind. 65(4), 675–692.

Kubler, S., Robert, J., Derigent, W., Voisin, A., Le Traon, Y.(2016). A State-of the-Art Survey & Testbed of FuzzyAHP (FAHP) applications. Expert Syst. Appl. 65, 398–422.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. B. (2016). Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. NIPS.

Laird, J., Rosenbloom, P., and Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism, Machine Learnisng, 1. 11-46, 1986. Reprinted in Buchanan, B. and Wilkins, D. (eds.), Readings in Knowledge Acquisition and Learning, Morgan Kaufmann, San Francisco, CA. 518-535

Lample G., Devendra, Sc. (2017). Playing Fps Games with Deep Reinforcement Learning. In Proceedings of the Thirty-first AAI Conference on Artificial Intelligence (Aaai'17). Aaai Press, 2140–2146.

Lane, H., Core, M., van Lent, M., Solomon, S., and Gomboc. D. (2005). Explainable Artificial Intelligence for Training and Tutoring. In Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology. IOS Press, Amsterdam, The Netherlands, The Netherlands, 762-764.

Lee, C. (2019). The Game of Go: Bounded Rationality and Artificial Intelligence. Iseas Yusof Ishak Institute. Http://hdl.handle.net/11540/10253.

Lee-Urban, S. (2018). Game AI, Finite State Machines. http://www.cc.gatech.edu/~surban6/2016-cs4731/.

Lei, X., Huang, A., Zhao, T., Su, Y. Chuan, R. (2018). A New Machine Learning Framework for Air Combat Intelligent Virtual Opponent. Journal of Physics: Conference Series. 1069. 012031. 10.1088/1742-6596/1069/1/012031.

Lieder, F., Krueger, P., Griffiths, T. (2017). An Automatic Method for Discovering Rational Heuristics for Risky Choice. (unpublished).

Leibo, J., Zambaldi,v., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems (Aamas '17). International Foundation for Autonomous Agents and Multiagent Systems, Richland, Sc, 464–473.

Lessiter, J., Freeman, J., Keogh, E. And Davidoff, J. (2001). A Cross-media Presence Questionnaire: the Itc-sense of Presence Inventory. Presence - Teleoperators and Virtual Environments 10(3): 282-297.

Lewis, R., Howes, A., Singh, S. (2014). Computational Rationality: Linking Mechanism and Behavior Through Bounded Utility Maximization. Topics in Cognitive Science, 6(2):279{311, 2014}.

Li, K. (2017). Learning to Optimize with Reinforcement Learning, Berkley Artificial Intelligence Research, https://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/

Li, X., Engelbrecht, a., and Epitropakis, M. G. (2013). Benchmark Functions for Cec'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization. Rmit University, Evolutionary Computation, and Machine Learning Group, Australia, Tech. Rep, 1-10. Liang, E. and Liaw, R. (2018). Scaling Multi-Agent Reinforcement Learning, The Berkeley Artificial Intelligence Research Blog, 4-5.

Lenham, M. (2018). Learn Unity ML-Agents Fundamentals of Unity Machine Learning, Packt Publishing.

Liarokapis, F. (2006). An Exploration from Virtual to Augmented Reality Gaming. Simulation & Gaming 37(4): 507-533.

Linick, M. E., Yurchak, J., Spirtas, M., Dalzell, S., Wong, Y., Crane, Y (2020). Hedgemony: A Game of Strategic Choices, Santa Monica, Calif.: RAND Corporation, TL-301-OSD, 2020. As of October 26, 2020: https://www.rand.org/pubs/tools/TL301.html

Littman. M. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. In Proceedings of the Eleventh International Conference on Machine Learning, 157–163.

Liu, Q., Wu, S., Wang, L. (2017). Multi-behavioral Sequential Prediction with Recurrent Log-bilinear Model. Ieee Trans. Knowl. Data Eng., 29(6):1254-1267. Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., Graepel, T. (2018). Emergent Coordination through Competition, DeepMind, London, United Kingdom.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Advances in Neural Information Processing Systems, 6379–6390.

Luan, S., Schooler, L. J., & Gigerenzer, G. (2011). A Signal-Detection Analysis of Fast-and-Frugal Trees. Psychological Review, 118(2), 316–338.

Lundberg, S., Su-in, L. (2017). A Unified Approach to Interpreting Model Predictions, Advances in Neural Information Processing Systems 30, Curran Associates, Inc, 4765-4774.

Lyu, D., Yang, F., Liu, B., and Gustafson, S. (2019). SDRL: Interpretable and Data-efficient Deep Reinforcement Learning Leveraging Symbolic Planning. In AAAI 2019.

Malpas, J. (2012). The Stanford Encyclopedia of Philosophy (Winter Edition 2012), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/entries/bounded-rationality/>.

Privosnik, M. Marolt, a. Kavcic, and S. Divjak (2002). Evolutionary Construction of Emergent Properties in Multi-Agent Systems. 327–330.

Magnenat-Thalmann, N. and Bonanni, U. (2006). Haptics in Virtual Reality and Multimedia. IEEE Multimedia: 6-11.

Mannor, S. Simester, D., Sun, P., Tsitsiklis, J. (2007). Bias and Variance Approximation in Value Function Estimates. Management Science, 53(2):308–322.

Manson, Steven. (2006). Bounded Rationality in Agent-based Models: Experiments with Evolutionary Programs. International Journal of Geographical Information Science. 20.

Martignon, L., Vitouch, O., Takezawa, M., Forster, M. R. (2003). Naive and Yet Enlightened: From Natural Frequencies to Fast-and-frugal Decision Trees. Thinking: Psychological Perspective on Reasoning, Judgment, and Decision-making, 189–211.

Martignon, L., Katsikopoulos, K. V., Woike, J. K. (2008). Categorization with Limited Resources: A Family of Simple Heuristics. Journal of Mathematical Psychology, 52(6), 352–361.

Mata, R., Thorsten, P., Von Helversen, B., Hertwig, R., Rieskamp, J., Schooler,L. (2012). Ecological Rationality: A Framework for Understanding and Aiding the Aging Decision Maker, Frontiers in Neuroscience, 19.

Matignon, L., Laurent, G., Fort-piat, N. (2012). Independent Reinforcement Learners in Cooperative Markov Games: A Survey Regarding Coordination Problems. The Knowledge Engineering Review. 27, 1-31.

Matiisen, T. (2015). Demystifying Deep Reinforcement Learning. Neuro.cs.ut.ee. Computational Neuroscience Lab. Retrieved 2018-04-06.

Melo, F.s., & Ribeiro, M. (2007). Q-Learning with Linear Function Approximation. Colt.

Milgram, P., Haruo T., Akira U., and Fumio K. (1994). Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. SPIE 2351 282–92.

Mill, J.S. (1848)/ Principles of Political Economy with Some of their Applications to Social Philosophy, 1 (1 ed.), London: John W. Parker, retrieved 7 December 2012, volume 2.

Miller, J., and Page. S. (2007). Complex Adaptive Systems: An Introduction to Computational Models of Social Life. Princeton University Press, Full article.

Miller, T (2019). Explanation in Artificial Intelligence: Insights from the Social Sciences. Artificial Intelligence 267 (2019): 1–38.

Mirshekarian, S., &Sormaz, D. N. (2018). Machine Learning Approaches to Learning Heuristics for Combinatorial Optimization Problems. Procedia Manufacturing. 17. 102-109. 10.1016/j.promfg.2018.10.019.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, a., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. Nips Deep Learning Workshop 2013.

Mnih, V., Badia, A., Mirza, M., Graves, A. Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. Unpublished.

Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012). Foundations of Machine Learning, The MIT Press ISBN 9780262018258.

Mousavi, S. (2017). Heuristics are Tools for Uncertainty, Homo Oeconomicus, Vol 34, 4,

Muggleton, S. (1991). Inductive Logic Programming, New Generation Computing, 8. 295-318.

Murphy, K. P. (2012). Machine Learning - A Probabilistic Perspective. Adaptive Computation and Machine Learning Series. MIT Press.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K. and Silver, D. (2015). Massively Parallel Methods for Deep Reinforcement Learning. Arxiv Preprint. arXiv: 1507. 04296.

Navarro-Martinez, D., Loomes, G., Isoni, a., Butler, D., Alaoui, L. (2018). Boundedly Rational Expected Utility Theory, Journal of Risk and Uncertainty, Springer, Vol. 57(3), Pages 199-223, December.

Neto, G. (2005). From Single-Agent to Multi-Agent Reinforcement Learning: Foundational Concepts and Methods.

Neto, G., Lima, P. (2005). Minimax Value Iteration Applied to Robotic Soccer. In: IEEE ICRA 2005 Workshop on Cooperative Robotics. Barcelona, Spain, 1-4.

Niazi, M., Hussain, A. (2011). Agent-based Computing from Multi-agent Systems to Agent-Based Models: A Visual Survey (PDF). Scientometrics. 89 (2): 479–499.

Nils J. Nilsson. (1997). Artificial Intelligence: A New Synthesis. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Full article.

Nilsson, Nils J. (2010). The Quest for Artificial Intelligence: A History of Ideas and Achievements. Cambridge: Cambridge University Press.

Nischelwitzer, A., Lenz, F. J., Searle, G., &Holzinger, A. (2007). Some Aspects of the Development of Low-Cost Augmented Reality Learning Environments as Examples for Future Interfaces in Technology Enhanced Learning. In C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction. Applications and Services (728-737). Berlin: Springer.

Norman, D. A., and Bobrow, D. G. (1975). On Data-limited and Resource-limited Processes. Cogn. Psychol. 7, 44–64. Doi: 10.1016/0010-0285(75)90004-3.

Russel, S., Norvig., P. (2010). Artificial Intelligence: A Modern Approach, Third Edition. New Jersey: Prentice Hall.

Nowak M. (2006). Five Rules for the Evolution of Cooperation. Science.314(5805):1560–1563.

Oh, J., Chockalingam, V., Singh, S., & Lee, H. (2016). Control of Memory, Active Perception, and Action in Minecraft, Arxiv Preprint Arxiv:1605.09128.

Ohtsuki H., Hauert C., Lieberman E., Nowak J. (2006). A Simple Rule for the Evolution of Cooperation on Graphs and Social Networks. Nature. 441(7092):502–505.

Oikonomopoulos, A., Pantic, M. (2013). Human Activity Recognition Using Hierarchically-Mined Feature Constellations, 150–159.

Ortega, P., Lee, D. (2014). An Adversarial Interpretation of Information-Theoretic Bounded Rationality. Proceedings of the National Conference on Artificial Intelligence. 4.

Osoba, O., Vardavas, R., Grana, J., Zutshi, R., &Jaycocks, A. (2020). Policy-focused Agent-based Modeling using RL Behavioral Models. ArXiv, abs/2006.05048.

Oudeyer, P. Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic Motivation Systems for Autonomous Mental Development. Evolutionary Computation, IEEE Transactions on, 11, 265–286.

Özbakır, A. Baykasoğlu, and S. Kulluk. (2010). A Soft Computing-Based Approach for Integrated Training and Rule Extraction from Artificial Neural Networks: DIFACONN-miner, Applied Soft Computing, Vol. 10, no. 1. 304–317.

Pals, G. (2018). Opening the Black Box of Machine Learning: Let's See What's Happening. Medium publishing, https://tinyurl. com/y32de6te

Panait, L., Luke., S. (2005). Cooperative Multi-Agent Learning: The State of the Art. In: Autonomous Agents and Multi-Agent Systems 11. 3. 387–434.

Papadimitriou, C., and Tsitsiklis, J. (1987). The Complexity of Markov Decision Processes. Mathematics of Operations Research 12(3):441–450.

Papoudakis, G., Christianos, F., Rahman, a., & Albrecht, S.V. (2019). Dealing with Non-stationarity in Multi-Agent Deep Reinforcement Learning. Arxiv, Abs/1906.04737.

Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., & Rohrbach, M. (2018). Multimodal Explanations: Justifying Decisions and Pointing to the Evidence.

Parr, R., Russell, S. (1998). Reinforcement Learning with Hierarchies of Machines, In Advances in Neural Information Processing Systems Vol 10, 1043-1049 Cambridge MA MIT Press.

Partalas, I., Feneris, I., Vlahavas, I. (2007). Multi-agent Reinforcement Learning Using Strategies and Voting. In 19th IEEE International Tools on Artificial Intelligence, 318–324.

Pashevich, A., Hafner, D., Davidson, J., Sukthankar, R., & Schmid, C. (2018). Modulated Policy Hierarchies. arXiv preprint arXiv:1812.00025.

Pathak, D., Agrawal, P., Efros, A. A., Darrell, T. (2017). Curiosity-Driven Exploration by Self-Supervised Prediction, 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, 2017, 488-489.

Perez-Liebana, D., Hofmann, K., Prasanna S., Kuno, N., Kramer, A., Devlin, S., Gaina, R. (2018). The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ). Competition, 2019, Challenges in Machine Learning (NIPS Workshop). 1-4.

Persky, J. (1995). Retrospectives: The Ethology of Homo Economicus. The Journal of Economic Perspectives, Vol. 9, No. 2 (Spring, 1995), 221–231.

Peters, J.; Mulling, K.; and Altun, Y. (2010). Relative Entropy Policy Search. In AAAI.

Puterman, Martin L. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. Machine Learning, 5:239-266.

Rand, W. (2017). Machine Learning Meets Agent-Based Modeling: When Not to Go to a Bar. https://ccl.northwestern.edu/papers/agent2006rand.pdf

Resnick, M. (1997). Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds (Complex Adaptive Systems). The MIT Press, January.

Rhodes, C. J., Anderson, R. M. (1996). Power Laws Governing Epidemics in Isolated Populations, Nature, 381, 600–602.

Ribeiro, M.T., Singh, S., Guestrin, C. (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 (KDD '16). Association for Computing Machinery, New York, NY, USA, 1135–1144.

Ros, R., LluisArcos, J., Lopez R. Veloso. M. (2009). A Case-Based Approach for Coordinated Action Selection in Robot Soccer. Artificial Intelligence, 173(9-10):1014–1039.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 65(6):386.

Roy, A., Xu, H., Pokutta. S. (2017). Reinforcement Learning under Model Mismatch. 31st Conference on Neural Information Processing Systems.

Rudd, D. M. (2010). The Effects of Heuristic Problem-Solving Strategies on Seventh Grade Students' Self-Efficacy and Level of Achievement in Mathematics.

Russell, S. (1997). Rationality and intelligence. Artificial Intelligence, 94:57–77,1997.

Russell, S.J., Norvig, P. (2003). Artificial Intelligence: A Modern Approach, Prentice-Hall, New York, NY, 1080.

Ryan, M. J. (1988). Constraints and Patterns in the Evolution of Anuran Acoustic Communication. In B. Fritzsch, T. Hetherington, M.J. Ryan, W. Walkowiad& W. Wilczynski (Eds.). The evolution of the amphibian auditory system. New York, NY: Wiley, John & Sons.637–677.

Saad, E., Wunsch, D. (2007). Neural Network Explanation Using Inversion, Neural Networks, Vol. 21. 78-93.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210-229.

Saroj, K. A. (2009). Decision-Making: Meaning and Definition. Retrieved from http://www.excellentguru.com/index. php.

Saaty, T.L. (1999). Decision-Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World. RWS Publications, Pittsburgh.

Sato, M., Tsukimoto, H. (2001). Rule Extraction from Neural Networks via Decision Tree Induction, in International Joint Conference on Neural Network, Washington, DC, 2001. 1870 - 1875 Vol. 3.

Scheibehenne, B., Miesler, L., & Todd, P. M. (2007). Fast-and-frugal Food Choices: Uncovering Individual Decision Heuristics. Appetite, 49, 578–589.

Schelling TC. (1971). Dynamic Models of Segregation. Journal of Mathematical Sociology, 1(2):143–186.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust Region Policy Optimization. In Blei, D., and Bach, F., editors, Proceedings of the 32nd International Conference on Machine Learning (ICML-15). 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, a. &Klimov, O. (2017). Proximal Policy Optimization Algorithms. Corr, Abs/1707.06347.

Shapley, L S. (1953). A Value for N-Person Games.Contributions to the Theory of Games 2.28 (1953): 307-317.

Sen, S., Sekaran, M., Hale, J. (1994). Learning to Coordinate without Sharing Information. In: Proceedings of the 13th National Conference on Artificial Intelligence.

Sen, S., Weiss, G. (1999). Learning in Multiagent Systems. In: G. Weiss (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, chap. 6. 259–298. MIT Press.

Sert, E., Bar-Yam, Y. & Morales, A.J. (2020). Segregation Dynamics with Reinforcement Learning and Agent-based Modeling. Sci Rep 10, 11771.

Setiono, R. (2000). Extracting M-of-N Rules from Trained Neural Networks, IEEE Transactions of Neural Networks, Vol. 11, no. 2. 512-519.

Setiono, R., Liu, H. (1995). Understanding Neural Networks via Rule Extraction, In Proceedings of the International Joint Conference on Artificial Intelligence, 480-485.

Setiono, R., Liu, H. (1996). Improving Backpropagation Learning with Feature Selection, Applied Intelligence, Vol. 6, no. 2, 129-140.

Shelton, C. (2000). Balancing Multiple Sources of Reward in Reinforcement Learning. Advances in Neural Information Processing Systems.

Sherman, W., Craig, A. (2002). Literacy in Virtual Reality: A New Medium, IEEE Virtual Reality Conference, p. 183, March 24-28, 2002. 81.

Sherwood, T., Calder, B. (2001). Automated Design of Finite State Machine Predictors for Customized Processors. In Proceedings of the 28th annual international symposium on Computer architecture (ISCA ’01). Association for Computing Machinery, New York, NY, USA, 86–97.

Shoham, Y., Leyton-Brown,K. (2008). Multiagent Systems Algorithmic, Game Theoretic and Logical Foundations. Cambridge University Press.

Shrikumar, A., Greenside,P., Kundaje. A. (2017). Learning important features through propagating activation differences. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17). JMLR.org, 3145–3153.

Simon, Herbert A. (1947). Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization, first edition, New York: Macmillan.

–––, 1955a, “A Behavioral Model of Rational Choice”, Quarterly Journal of Economics, 69(1): 99–118. doi:10. 2307/1884852.

–––, 1955b, “On a Class of Skew Distribution Functions”, Biometrika, 42(3–4): 425–440. doi:10. 1093/biomet/42. 3-4. 425.

–––, 1957a, Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization, second edition, New York: Macmillan.

–––, 1957b, Models of Man, New York: John Wiley.

–––, 1969, The Sciences of the Artificial, 3rd ed. MIT Press.

–––, 1972, Theories of Bounded Rationality. In Radner, C., and Radner, R., Eds., Decision and Organization. Amsterdam: North Holland Publ. 161–176.

–––, 1976, From Substantive to Procedural Rationality, in 25 Years of Economic Theory, T. J. Kastelein, S. K. Kuipers, W. A. Nijenhuis, and G. R. Wagenaar (eds.), Boston, MA: Springer US, 65–86. doi:10. 1007/978-1-4613-4367-7_6.

–––, 1990, Utility and Probability. Palgrave Macmillan UK, Eatwell, J. and Milgate, M. and Newman, P.

Skinner, B.F. (1938). The Behavior of Organisms: An Experimental Analysis. Appleton-Century.

Smola, A., Vishwanathan, S. V. N. (2008). Introduction to Machine Learning, Purdue University Press, 42-84.

Snook, B., Taylor, P. J., &Bennel, C. (2004). Geographic Profiling: The Fast, Frugal, and Accurate Way. Applied Cognitive Psychology, 18, 105–121.

Snook, B., Zito, M., Bennell, C., & Taylor, P. J. (2005). On the Complexity and Accuracy of Geographic Profiling Strategies. Journal of Quantitative Criminology, 21, 1–26.

Squazzoni, F. (2012). Agent-Based Computational Sociology, John Wiley & Sons, Ltd, 10-11.

Stanney, K. M. (Ed.) (2002). Handbook of Virtual Environments: Design, Implementation, and Applications. Mahwah, NJ: Lawrence Erlbaum Associates.

Stefanidis, A., Jenkins A., Croitoru, A. and Crooks, A. T. (2016). Megacities Through the Lens of Social Media, Journal of the Homeland Defense & Security Information Analysis Center (HDIAC), 3(1): 26-28.

Stigler, G. J. (1961). The Economics of Information. Journal of Political Economy, 69, 213-225.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems

(NIPS'99). S. A. Solla, T. K. Leen, and K. Müller (Eds.). MIT Press, Cambridge, MA, USA, 1057-1063.

Sutton, R. and Barto, A. (1995). Reinforcement Learning: An Introduction. MIT Press.

Sutton, R. and Barto, A. (1998). Reinforcement Learning, 2nd Edition: An Introduction. MIT Press.

Swartout, W. R., Paris, C. L., and Moore, J. D. (1994). Design for Explainable Expert Systems. IEEE Expert. 6, Number 3, 58-64.

Taha, A., Ghosh, J. (1999). Symbolic Interpretation of Artificial Neural Networks, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, no. 3. 448–463.

Tampuu, A., Matiisen T., Kodelja, D., Kuzovkin, We., Korjus, K., Aru, J., Aru, J., and Vicente, R. (2017). Multi-agent Cooperation and Competition with Deep Reinforcement Learning. PloS one.

Tan. A. (1993). Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In Proc. 10th International Conference on Machine Learning.

Tan, Y., Xu, X., Liu. Y. (2016). Improved Recurrent Neural Networks for Session-based Recommendations. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016). Association for Computing Machinery, New York, NY, USA, 17–22.

Terry, J.K., &Grammel, N. (2020). Multi-Agent Informational Learning Processes.

Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. In: Advances in Neural Information Processing Systems, (John E. Moody, Steve J. Hanson, and Richard P. Lippmann, Eds.). Vol. 4. Morgan Kaufmann Publishers, Inc. 259–266.

Tesauro, Gerald (1992). Practical Issues in Temporal Difference Learning. In: Advances in Neural Information Processing Systems (John E. Moody, Steve J. Hanson and Richard P. Lippmann, Eds.). Vol. 4. Morgan Kaufmann Publishers, Inc. 259–266.

Thorndike, E. L. (1911). Animal Intelligence. New York: Macmillan (Reprinted Bristol: Thoemmes.

Thrun S. (1995). Extracting Rules from Artificial Neural Networks with Distributed Representations. Advances in Neural Information Processing Systems. 505, 12.

Todd, P., Gigerenzer, G. (2007). Environments That Make Us Smart. 16. 167-171. 10.1111/j.1467-8721.2007.00497. Oxford University Press, 2012.

Todd, P.M., Brighton, H. (2016). Building the Theory of Ecological Rationality. Minds & Machines 26, 9–30. https://doi.org/10.1007/s11023-015-9371-0

Torrey, L., Walker, T., Shavlik, J., Maclin, R. (2005). Using Advice to Transfer Knowledge Acquired in One Reinforcement Learning Task to Another. In: Gama J., Camacho R., Brazdil P.B., Jorge A.M., Torgo L. (eds) Machine Learning: ECML 2005. ECML 2005. Lecture Notes in Computer Science, vol 3720. Springer, Berlin, Heidelberg.

Tsukimoto, H. (2000). Extracting Rules from Trained Neural Networks. Neural Networks, IEEE Transactions on, 11(2):377–389.

Tuomi, I. (2018). The Impact of Artificial Intelligence on Learning, Teaching, and Education: Policies for the Future. 10.2760/12297.

van der Hoog, S. (2016). Deep Learning in Agent-Based Models: A Prospectus, arXiv:1706. 06302.

Von Neumann, J., Oskar, M. (1953). Theory of Games and Economic Behavior. Princeton, Nj. Princeton University Press.

Vu, T., Probst, C., & Epstein, J., Brennan, A., Strong, M., Robin, P. (2019). Toward Inverse Generative Social Science Using Multi-Objective Genetic Programming. 1356-1363.

Wallace, R., Geller, A., Ayano, V (2015). Assessing the Use of Agent-Based Models for Tobacco Regulation, The National Academies Press, Washington, DC, Full article

Wang, H., Wu, X., Sun, L., & Du, B. (2019). Passenger Behavior Prediction with Semantic and Multi-pattern LSTM Model. IEEE Access, 7, 157873-157882.

Watkins, C. J. C. H. (1989). Learning with Delayed Rewards. Ph. D. Dissertation, Cambridge University.

Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. Ph.D. Thesis, Cambridge University, Cambridge, England.

Weick, K. (1979). The Social Psychology of Organization (2 ed.). New York: McGraw Hill.

Werker, C. and Brenner, T. (2004). Empirical Calibration of Simulation Models. Papers on Economics and Evolution 2004-10, Philipps University Marburg, Department of Geography.

Wilensky, U. (1997). NetLogo Carnivore Herbivore Predation Model. http://ccl. northwestern. edu/netlogo/models/Carnivore Herbivore Predation. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. (1999). NetLogo. http://ccl. northwestern.edu/Netlogo/.Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Williams, Ronald J. (1987). A Class of Gradient-Estimating Algorithms for Reinforcement Learning in Neural Networks. Proceedings of the IEEE First International Conference on Neural Networks.

Wong, K. C. (2015). Evolutionary Multimodal Optimization: A Short Survey. arXiv preprint arXiv:1508. 00457.

Wu, Y., & Tian, Y. (2016). Training Agent for First-person Shooter Game with Actor-Critic Curriculum Learning. ICLR 2017 Conference.

Yildizoglu, M. and Salle, I. (2012). Efficient Sampling and Metamodeling for Computational Economic Models. Cahiers du GREThA 2012-18, Groupe de Recherche enEconomieTheorique et Appliquee.

Yining, W., Yuxian, J. (2003). An Intelligent Differential Game on Air Combat Decision Flight Dynamics 21 66-70.

Zargarpour, H., H. LaBounta, et al. (2010). Interactive Games. The Ves Handbook of Visual Effects: Industry Standard Vfx Practices and Procedures: 707-736. 1st edition. Publisher: Focal Press. ISBN-13: 9780240812427. ISBN-10: 0240812425.

Zeng, Z., Miao, C, Leung,C. Chin, J. J. (2018). Building More Explainable Artificial Intelligence with Argumentation AAAI Publications, Thirty-Second AAAI Conference on Artificial Intelligence.

Zhang, C. Bengio, S. Hardt, M. Recht, B. Vinyals, O. (2017). Understanding Deep Learning Requires Rethinking Generalization," in ICLR 2017.

Zhang, L., Zhang, B. (2006). Hierarchical Machine Learning – A Learning Methodology Inspired by Human Intelligence, Lecture Notes in Computer Science, Conference: Rough Sets and Knowledge Technology, First International Conference, RSKT, Chongqing, China.

Zheng, H., Jiang, J., Wei, P., Long, G., Zhang, C. (2020). Competitive and Cooperative Heterogeneous Deep Reinforcement Learning. In Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (Aamas 2020), Auckland, New Zealand, May 9–13,2020, Ifaamas, 9.

Zhou, Z. H., Chen, S. F., and Chen, Z. Q. (2000). A Statistics-based Approach for Extracting Priority Rules from Trained Neural Networks. In Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-inns-enns International Joint Conference on, 1).3, 401–406. IEEE.

Zilke, J. (2015). Extracting Rules from Deep Neural Networks, M. S. Thesis, Computer Science Department, Technische Universität Darmstadt.

Zintgraf, L. M., Cohen, T. S., Adel, T., & Welling, M. (2017). Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. CoRR, abs/1702. 04595. Retrieved from http:// arxiv. org/abs/1702. 04595.

BIOGRAPHY

Paul Cummings graduated from Wethersfield High School, Wethersfield, CT in 1987. He received his Bachelor of Arts from Boston University in 1992. In his many leadership roles, he has developed a career in research, technology, and management. He received a Master of Interdisciplinary Studies from George Mason University in 2018.