ABSTRACT

Title of Dissertation:	Towards Trust and Transparency in Deep Learning Systems through Behavior Introspection & Online Competency Prediction
	Julia Filiberti Allen, Doctor of Philosophy, 2021
Dissertation directed by:	Dr. Steven A. Gabriel, Department of Mechanical Engineering

Deep neural networks are naturally "black boxes", offering little insight into how or why they make decisions. These limitations diminish the adoption likelihood of such systems for important tasks and as trusted teammates. We employ introspective techniques to abstract machine activation patterns into human-interpretable strategies and identify relationships between environmental conditions (why), strategies (how), and performance (result) on both a deep reinforcement learning two-dimensional pursuit game application and image-based deep supervised learning obstacle recognition application. Pursuit-evasion games have been studied for decades under perfect information and analytically-derived policies for static environments. We incorporate uncertainty in a target's position via simulated measurements and demonstrate a novel continuous deep reinforcement learning approach against speed-advantaged targets. The resulting approach was tested under many scenarios and performance exceeded that of a baseline course-aligned strategy. We manually observed separation of learned pursuit behaviors into strategy groups and manually hypothesized environmental conditions that affected performance. These manual observations motivated automation and abstraction of conditions, performance and strategy relationships. Next, we found that deep network activation patterns could be abstracted into human-interpretable strategies for two separate deep learning approaches. We characterized machine commitment by the introduction of a novel measure and revealed significant correlations between machine commitment, strategies, environmental conditions, and task performance. As such, we motivated online exploitation of machine behavior estimation for competency-aware intelligent systems. And finally, we realized online prediction capabilities for conditions, strategies, and performance. Our competency-aware machine learning approach is easily portable to new applications due to its Bayesian nonparametric foundation, wherein all inputs are compactly transformed into the same compact data representation. In particular, image data is transformed into a probability distribution over features extracted from the data. The resulting transformation forms a common representation for comparing two images, possibly from different types of sensors. By uncovering relationships between environmental conditions (why), machine strategies (how), & performance (result) and by giving rise to online estimation of machine competency, we increase transparency and trust in machine learning systems, contributing to the overarching explainable artificial intelligence initiative.

TOWARDS TRUST AND TRANSPARENCY IN DEEP LEARNING SYSTEMS THROUGH BEHAVIOR INTROSPECTION & ONLINE COMPETENCY PREDICTION

by

Julia Filiberti Allen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Dedication

To my family.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119S0030, known as the MindfuL™ program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. Approved for public release, unlimited distribution. Not export controlled per ES-FL-051121-0060.

For Chapter 2, I acknowledge collaboration with Steve Schmidt for the reinforcement learning implementation and PyTorch support. I also acknowledge consultation from Dr. Lucas Finn on the target tracker implementation. For Chapter 3, I acknowledge support from Steve Schmidt for PyTorch and dimension reduction support. All of the design, experimentation, and analysis in Chapters 2 and 3 was my own work; additionally, all but the base reinforcement learning deep deterministic policy gradient harness software development work was my own.

And for Chapter 4, I acknowledge the contributions of a large research team across BAE Systems FAST Labs™ and the MIT Computer Science & Artificial Intelligence (CSAIL) department, for which I served as Principal Investigator (PI) and lead proposal author. Specifically, I acknowledge Dr. Michael Planer, Dr. Olga Babko-Malaya, Myriam Ayad, Collin Blakely, Christopher Dean, Dr. John Fisher III, Genevieve Flaspohler, Dr. Matthew Henry, Jennifer Hemingway, Dr. Steven Kalik, Dr. Eric Liu, Jennifer Sierchio, Steve Schmidt, Dr. Sean Stromsten, and Dr. Elizabeth Wills for their technical contributions and project support on the MindfuL™ program. As PI, I was responsible for the design, implementation oversight, delivery, and presentation associated with the program. I was the lead author of the original

MindfuL™ winning DARPA proposal, and the ideas for all of the implemented components and their interactions were my own. I was the lead system architect and held the final implementation decision authority for all component development. I designed all of the components and their interactions with one another to achieve a competency-aware *system*. Moreover, I implemented the prototype Environment Similarity Calculator component on my own. And finally, the writing of all parts of this dissertation were my own, with edit suggestions from my advisor, Dr. Steven A. Gabriel. In addition to the MindfuL™ team, I acknowledge program guidance and technical direction from DARPA, especially Dr. Jiangying Zhou and Dr. Zachary Lapin, whose interpretations and research interests around machine competency inspired the MindfuL™ software development from inception to implementation. Throughout this dissertation, I used "we" to acknowledge contributions from others, however, the ideas behind all research included in this dissertation were my own.

Finally, I acknowledge the impactful guidance, countless reviews, and valuable feedback from my advisor, Dr. Steven A. Gabriel. His dedication and commitment to my research amidst the COVID-19 pandemic and ever-fluctuating full-time employment & military commitments are greatly and sincerely appreciated.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

alternative asynchronous actor-critic (A3C)
artificial intelligence (AI)
competency-aware machine learning (CAML)
Computer Science & Artificial Intelligence (CSAIL)
Defense Advanced Research Projects Agency (DARPA)
deep discrete policy gradient (DDPG)
Deterministic Policy Gradient (DPG)
eXplainable Artificial Intelligence (XAI)
global positioning system (GPS)
hierarchical Dirichlet process (HDP)
inertial measurement unit (IMU)
Kullback-Leibler divergence (KL-divergence)
light detection and ranging (LiDAR)
machine learning (ML)
Markov Decision Process (MDP)
meters (m)
meters per second (m/s)
mixed complementarity problem (MCP)
Modified National Institute of Standards and Technology (MNIST)
National Security Commission on Artificial Intelligence (NSCAI)
Naval Applications for Machine Learning (NAML)
Oriented FAST and Rotated BRIEF (ORB)
Principal Investigator (PI)
Principle Component Analysis (PCA)
Q-Learning fuzzy inference system (QFIS)
receiver operating characteristic (ROC)
reinforcement learning (RL)
Robot Operating System (ROS)
Shapley Additive exPlanations (SHAP)
t-distributed Stochastic Neighbor Embedding (t-SNE)
two-dimensional (2-d)
Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)
unmanned systems (UxV)
unmanned aerial vehicle (UAV)
unmanned underwater vehicles (UUV)
unmanned surface vessels (USV)
unmanned ground vehicles (UGV)
user interface (UI)
Value of Information (VoI)

Chapter 1: Introduction

Section 0: Motivation

A strong motivation for autonomous control and XAI research is the explosion of unmanned systems (UxV). First came militarization and commercialization of unmanned aerial vehicle (UAV) proliferation; now comes amassing of small satellites, unmanned underwater vehicles (UUV), unmanned surface vessels (USV), and unmanned ground vehicles (UGV). To date, deployed UxV autonomy applications have been largely limited to point-to-point autonomy and subsystem control, leaving gaps in higher intelligence and decision-making skills for tasks with higher complexity. One task which has been studied at depth for simplified environments is the pursuit game, wherein one agent attempts to capture a target agent, and the extension of the game, called a pursuit-evasion game, where the target agent actively evades the pursuer agent.

Many applications for pursuit games arise naturally in unmanned systems control, air-to-air combat, ballistic missile defense, and sports. In unmanned systems control, agents plan routes to navigate toward waypoints and avoid obstacles. Additionally, unmanned agents may seek to rendezvous with teammates for refueling, recharging, leader-follower convoys, or other activities. In air-to-air combat, systems may aim to gain stable line-of-sight with an opposing system. In missile defense, autonomous systems may seek to intersect the path of an incoming threat. In sports, players attempt to intercept passes, make tackles, or catch a ball by anticipating future target object locations in dynamic environments. While some of these applications require three-dimensional modeling, in Chapter 2 we consider a two-dimensional (2-

d) version of this widely applicable game that clearly illustrates the value of the proposed continuous reinforcement learning (RL) approach. 2-d pursuit games apply to unmanned ground vehicle and unmanned surface vessel applications and are easily comparable to the vast literature in pursuit-evasion games.

Explainability, interpretability, and competency awareness are widely recognized shortcomings of current artificial intelligence (AI) and machine learning (ML) systems [1] [2]. Currently, many AI-system end-users lack trust and interest in the adoption of AI systems. They demand more explanations and rationale that support machine-derived solutions. As such, explainable artificial intelligence has been identified as priority research areas by the National Security Commission on Artificial Intelligence (NSCAI) [3] and funded accordingly by DARPA [4].   In this research, we explore ways to apply existing techniques to qualitatively improve trust and transparency in machine learning systems. The resulting capabilities contribute to the overarching field of explainable AI through analysis of machine behaviors and insights gained from making connections between behaviors and performance. No prior XAI research efforts have attempted to aggregate time-series machine behaviors into strategies that are relatable to humans. And no prior research has attempted to estimate environmental conditions, strategies, and performance using online predictors to provide a competency-aware machine learning system. We focus on the machine side of human-machine teaming, where we equip the machine with self-awareness, including the assessment of its own competency.

Section 1: Literature Survey

Pursuit-Evasion Games, like the Homicidal Chauffeur Problem, have been studied since introduced by Rufus Isaacs in 1951 [5], most commonly under perfect information and analytically-derived policies for static-game parameters, like constant-pursuer and evader speeds [6] [7]. The Homicidal Chauffeur Problem is a differential pursuit-evasion game in which a low-speed agent with an infinitely small turning radius, like that of a pedestrian, evades a much faster pursuer with a constrained-turning radius, like that of a chauffeur driving a car. Through calculus of variations and level-set methods, optimal control laws are derived for all combinations of speed and maneuverability ratios of the pursuer to the target [8]. In [9], the authors modeled an uncertain pursuit-evasion game using uncertain differential equations and derived a solution via the corresponding Riccati equation. And in [10], the authors used mixed complementarity problem (MCP) formulation to address a pursuit-evasion game amidst obstacles and uncertainty.

Recently, machine-learning approaches have been applied to develop robust strategies for pursuit-evasion and pursuit-only games. Such approaches seek to expand capture probabilities of success in cases where speed and maneuverability ratios are not suitable for applying the analytically derived solutions in the works surveyed in [6] and [7]. For example, these methods solve for optimal policies for pursuit-evasion games where both players have perfect information about one another and both players act rationally (optimally). Because of these assumptions, there are no solutions for solving a game where the evader moves faster than the pursuer [8], as the evader would move in a direction opposite of the pursuer’s direction (known with certainty) and the pursuer would never be able to catch up to the evader due to the speed disadvantage.

In [11], the authors train pursuit and evader agents under a reinforcement-learning approach which operates under constant speeds and position certainty. This was the only reference found which applied a continuous RL approach to a pursuit game, the focus of Chapter 2. In [12], the author trains only the pursuer to learn the homicidal chauffeur strategy using a two-stage, learning technique combining particle-swarm, optimization-based fuzzy logic controller algorithm with the Q-Learning fuzzy inference system (QFIS) algorithm to tune the parameters of a fuzzy logic controller. Similarly, in [13], the authors present a technique to tune a pursuer fuzzy logic controller using Q(λ)-learning and a genetic algorithm. In [14], multiple pursuer agents were trained using Watkin's Q(λ)-learning algorithm to successfully capture a single stationary target, but the algorithm did not extend well to moving target scenarios. Briefly, Q-Learning is a reinforcement learning approach that determines an optimal control policy by maximizing the expected reward from the current state until the end of the game (Chapter 2 Section 3). In our pursuit game, our control policy consists of heading and acceleration pursuit agent actions (Chapter 2.3), our state consists of observations available to the pursuer agent (Chapter 2 Section 2.2), and our reward is a function of the distance between the pursuer and target agents and time (Chapter 2 Section 2.5).

While pursuit games are widely analyzed in these publications, none of these explore RL approaches to pursuit games with uncertain information as studied in Chapter 2 and promoted as an open research problem in the short survey in [6].

Additionally, none of the above research efforts considered a pursuit game where the pursuer and evader could accelerate. Lastly, a literature survey in pursuit-evasion games did not find any prior work studying the case where the evader could move faster than the pursuer. Tackling challenges of uncertainty, speed control, and speed overmatch in Chapter 2 support deployment of the resulting algorithm on real unmanned systems in dynamic environments. In this case, the pursuer agent must learn anticipatory strategies to capture speed-advantaged target and robust strategies to account for uncertainty.   And we see that the pursuer does learn anticipatory strategies inherent to its “L-shaped” & sweeping behaviors manually abstracted from winning scenarios in Chapter 2 and automatically abstracted into strategy groupings in Chapter 3. While prior reinforcement learning approaches have led to anticipatory behaviors over time, none have been abstracted automatically from activation patterns into explainable AI strategies before the work presented in Chapter 3. Prior work and the novel contributions of this dissertation to pursuit game research is summarized in Table 1.

Table 1 Novelty of Pursuit RL Research

Reference	**Players**	**RL**	**Uncertainty**	**Dynamic Speed Control**	**Speed Overmatch**	**Explainable AI Strategies**
Feng et al., 2018	2		✓			
Wang, Wang, & Yue, 2019	2	✓				
Al-Talabi & Schwartz, 2014	1	✓				
Desouky & Schwartz, 2009	1	✓				
Bilgin & Kadioglu-Urtis, 2015	>2	✓				
Chapter 2	**1**	✓	✓	✓	✓	
Chapter 3	**1**	✓	✓	✓	✓	✓

To date, eXplainable Artificial Intelligence (XAI) efforts have been largely focused on ante-hoc model design, feature importance and continuous system evaluation [15] [16]. Ante-hoc approaches, such as random forests and decision trees, incorporate explainability mechanisms into models themselves, enabling natural interpretability of results in terms of pre-defined features or conditions [17] [18]. Post-hoc methods, such as Shapley Additive exPlanations (SHAP) [19] and permutation methods, have also focused on feature importance for explainability [20] [21] while others have focused on understanding high-confidence failures and ambiguous results by stimulating examples [22]. In this case, *ante-hoc* methods refer to instrumentation of XAI capabilities prior to training of a deep learning system. By instrumenting explainable parameters that govern policy selection, ante-hoc methods provide explainability by design; at a rudimentary level, for example, if one employed a rule-based pursuit controller, an ante-hoc approach could bookkeep which rules (criteria) were satisfied and relay them along with the control policy. Analogously, random forest models, which are an ensemble of decision trees, split trees over branches that can be traced back for explainable results. Conversely, *post-hoc* methods deal with models that have already been trained, whether they carry natural interpretability via ante-hoc methods or not. We focus here on post-hoc models so that our XAI capabilities can be more broadly used and bolted on to existing machine learning frameworks without stipulations on how the system is trained. In other words, the approach outlined in Chapter 4 is applicable to any type of deep learning model for any type of data; it does not require ante-hoc explainability considerations or impose any special training specifications on the ML system under evaluation.

Importantly, few post-hoc XAI efforts have focused on understanding machine behaviors; *behaviors* are encoded directly into the activation values of neurons in deep networks. The activation patterns of the neurons themselves are representative of machine “thought processes”. Moreover, few XAI efforts have addressed time-series applications [23]. Awareness of machine behaviors provides insight into machine *competency*, which goes beyond characterization of machine performance [24] by abstracting machine behaviors into communicable strategies. While dimension-reduction techniques such as Principle Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) have been used to visualize activation patterns previously [25] [26] [27], none have employed the Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) technique, which shows significant benefits over t-SNE on time-series data [28] and is more accurate than PCA [29]. Furthermore, no XAI research efforts have attempted to aggregate time-series machine behaviors into generalizable groups, thereby abstracting machine behaviors into *strategies* that are relatable to humans. And finally, no research has attempted to estimate environmental conditions, strategies, and performance using online predictors to provide a competency-aware machine learning (CAML) system. Prior work and the novel contributions of this dissertation to XAI research is summarized in Table 2.

Table 2. Novelty of Competency-Aware Machine Learning XAI Research

Reference	Conditions	Strategies	Performance	Online Competency Prediction	Time-Series Strategies
Google, 2021	Feature importance, SHAP	-	Model training guidance, continuous evaluation	-	-
IBM, 2021	-	-	Training model and data selection focus, continuous evaluation	-	-
Krischnamurthy et al., 2021 & Schmidt et al., 2021	Feature importance, instrumentation	-	-	-	-
Hilton et al., 2020	Feature importance	Non-negative matrix factorization	Causality of conditions	-	-
Schubert et al., 2020	-	Feature Visualization	-	-	-
Booth et al., 2021	Generating stimulating conditions	-	High-confidence failures	-	-
Jaderberg et al., 2019	Manual	t-SNE	-	-	-
Zahavy & Mannor, 2016	Manual	t-SNE	Automated	-	-
Rauber, Fadel, & Falcao, 2017	Manual	t-SNE	Automated	-	-
Chapter 2	**Manual**	**Manual**	**Manual**	**-**	✓
Chapter 3	**Manual**	**UMAP**	**Automated**	**-**	✓
Chapter 4	**SHAP, HDP**	**UMAP**	**Automated**	✓	**-**

Section 2: Dissertation Organization & Research Community Involvement

The dissertation is organized as follows. Motivation and prior related research is summarized in Chapter 1 for both pursuit games and XAI research areas. In Chapter 2, we describe the design and implementation of a novel deep RL-based controller for pursuit games with uncertain information and speed-overmatched targets. In the game, a pursuer agent employs a deep deterministic policy gradient (DDPG) algorithm (Chapter 2 Section 3.2) [30] to capture a moving target under imperfect information. While pursuit games have been widely analyzed since 1951, none attempted an RL approach to pursuit games with uncertain information as we introduced in Chapter 2 and analyzed further in Chapter 3. Tackling challenges of uncertainty, speed control, and speed overmatch supports deployment of the resulting algorithm on real unmanned systems in dynamic environments. In this case, the pursuer must learn anticipatory strategies to capture speed-advantaged target and robust strategies to account for uncertainty. We found, in the cases where the target maximum speed is greater than that of the pursuer's, the pursuer is successful in a speed-overmatched game. This result is novel and no prior work has realized a RL pursuer agent capable of pursuing a speed-advantaged target. Moreover, we manually abstract RL behaviors into human-interpretable strategy groupings. We make observations between manually hypothesized conditions and machine performance, motivating further investigation into offline machine competency and online competency prediction. *Competency* refers to both the machine performance and the strategy that was employed; environmental conditions affect how the agent performs the task (strategy) and the associated performance. Offline competency understanding gives us insight into how conditions, strategies, and performance relate to one another. Online competency prediction allows us to take advantage of those insights and avoid failures or behaviors that are not desirable in the current situation; it also allows gives us an opportunity to perform the task manually or otherwise intervene to avoid a system from performing a task under conditions in which it has not yet been trained. Chapters 3 & 4 investigate machine competency in detail. Finally, in Chapter 2, we show that the deep RL-based controller outperformed a baseline control algorithm significantly overall (by attaining 100% more target captures) and across each manually hypothesized condition.

In Chapter 3, we uncover and analyze machine behaviors through dimension reduction and time-series clustering for an RL agent playing the two-dimensional pursuit game described in Chapter 2. We abstract machine behaviors into strategies automatically and assess effects of the same manually hypothesized conditions as found in Chapter 2 and discover that they align intuitively to automatically derived machine strategies. We define a novel measure of machine commitment and reveal relationships between commitment and machine performance. Interestingly, many of the strategies discovered automatically in Chapter 3 are consistent with those extracted manually in Chapter 2. Moreover, we make important observations across behaviors, game outcomes, environmental conditions, and human relatability. We demonstrate utility of machine introspection over action-only alternatives, uncover aggregate human-relatable strategies in terms of explainable initial conditions, and discuss how these relationships can be exploited for online performance prediction. Lastly, we define a novel measure of machine *commitment* based on the volatility of activation expressions, measured as a function of rolling-horizon Shannon entropy and analyze its correlation to resulting machine performance.

In Chapter 4, we extend strategy abstraction methods to a completely different application, one using a supervised deep learning approach to recognize obstacles in visual data (still images). Here, we demonstrate that the same automated strategy-abstraction techniques employed in Chapter 3 are generalizable to a different domain. To exploit the game outcome-strategy correlation identified in Chapter 3, we design an online strategy predictor to be used *in situ*, which enables a user-facing module that suggests when to trust the machine or recommend user intervention. Moreover, we devise a method, rooted in a Bayesian nonparametric (BNP) approach (Chapter 4 Section 2), for discovering conditions automatically that is generalizable to various input sources beyond images and pursuit game observations. After compactly describing the input data using the hierarchical Dirichlet processes (HDP) BNP approach (Chapter 4 Section 2.1), we layer on deep learning predictors and explainers to produce competency assessments of the underlying ML system in an online mode. Finally, we relay the competency assessment to the operator via a user interface at an update rate that exceeds user expectations. That is, the user can act on a near real-time competency assessment to preclude the ML system from incorrectly performing its task. For the pursuit game task, it could be an automatic controller that preserves energy by foregoing a low-likelihood capture. For the obstacle recognition task, it could be a human taking over the joystick for one of many forward-deployed UxS.

Now, we briefly summarize the relationships between the research in Chapters 1-5. In Chapter 1, we describe the need for increased trust and transparency in machine-learning systems in terms of environmental conditions (why), machine behavior (how), and machine performance (result). In Chapter 2, we manually hypothesize conditions and strategies that impact machine performance. In Chapter 3, we automatically abstract behaviors into strategy groupings that correspond to differing game outcomes (performance). In Chapter 4, we automatically extract conditions from environmental observations and implement online predictors for competency-controlling conditions, strategies, and performance for a supervised learning task. By studying two disparate deep learning techniques (reinforcement & supervised) along with two disparate tasks (track-based pursuit & image-based obstacle recognition), we supply widely-applicable capabilities for the XAI research community. The key insights and scientific contributions of this research are summarized in Table 3.

Table 3 Key Research Insights & Section References

Insight & Scientific Contribution	Section Reference
Developed control strategies using deep reinforcement learning, novel for speed-overmatch, speed control, and pursuit games with uncertainty.	Chapter 2 Sections 2 & 3
The resulting deep RL policy outperformed a baseline course-aligned strategy by 100% and does better with respect to harsher pursuit game conditions.	Chapter 2 Section 4
Manually abstracted machine behaviors into strategy groupings and discovered relationships between conditions and performance, exploitable for online competency prediction.	Chapter 2 Section 4.5
Neural network activation patterns were automatically abstracted into strategies using a novel procedure; some strategies naturally had human-interpretable meaning.	Chapter 3 Section 4
Dimension-reduced behaviors preserved more information relative to game outcomes than actions alone.	Chapter 3 Section 2
Automatically-derived strategies can be exploited as a predictor of performance.	Chapter 3 Section 4
A novel measure of machine commitment is significantly higher in winning pursuit games than losing pursuit games.	Chapter 3 Section 6
Bayesian nonparametric approach supports condition traceability, compact environment characterization, and is easily ported to new input data types.	Chapter 4 Section 2
Environment similarity calculation supports identification of untrained situations and potential sensor faults.	Chapter 4 Section 3
Prototype competency prediction capabilities significantly outperform random chance for strategy and performance prediction.	Chapter 4 Sections 5 & 6
Competency-aware machine learning approach is generalizable to a large number of applications and machine learning approaches, as evident by proof-of-concept experimentation on both a supervised learning & RL approach and image-based & pursuit game application.	Chapters 3 & 4

The work in Chapter 2 was presented at the Naval Applications for Machine Learning (NAML) conference in March 2021 (~30% acceptance rate) and tentatively accepted pending two sets of minor revisions to the *Military Operations Research Journal*. The work in Chapter 3 was submitted to the *Data Mining and Knowledge Discovery Special Issue on Explainable & Interpretable ML and Data Mining* in May 2021 (under review). The work in Chapter 4 is part of an ongoing $5 million / 3-year DARPA program named Competency-Aware Machine Learning (October 2019-September 2022), for which the author serves as Principal Investigator and winning proposal lead author. Preliminary results were presented to academic, industry, and government participants during the September 2020 Principal Investigator (PI) meeting. Additionally, a poster was presented at NAML 2020 & NAML 2021 conferences.

Chapter 2: Reinforcement Learning Approach to Speed-Overmatched Pursuit Games with Uncertain Target Information

Section 0 Overview

Pursuit-evasion games have been studied for decades under perfect information and analytically-derived policies for static environments. Differential equations are solved to directly obtain optimal game solutions. Here, we incorporate uncertainty in a target's position via simulated measurements and propose a continuous deep RL approach to support pursuit of a speed-advantaged target. An OpenAI gym environment was created for simulating pursuit game play. A Kalman filter was implemented for simulating data fusion of uncertainties associated with imperfect range and bearing measurements from the pursuer to the target. An actor-critic based, model-free, deep discrete policy gradient (DDPG) method [30] was implemented for incrementally training an agent to compete at a speed-overmatched pursuit game. Essentially, the algorithm aims to optimize the weights of two networks. The actor learns network weight parameters that produce "good" pursuer heading & acceleration actions based on state observation inputs. The critic learns network weight parameters that estimate the pursuit reward associated with state and action inputs.

Each of the DDPG descriptors has particular meaning, as described in Table 4.

Table 4 DDPG Method Definitions

DDPG Descriptor	**Short description**
actor-critic based	"Actor-critic" methods consist of two models; the actor model which gives an action for a given state and a critic model which anticipates the reward of a given action based on a given state. Our actor model gives a heading & acceleration action for each state observation (Section 2.2). Our critical model estimates the pursuer reward function (Section 2.5).
model-free	"Model-free" methods do not explicitly learn or leverage any known dynamics of the agent or the environment. For example, our deep learning approach does not explicitly account for the limited turning radius of the pursuer. However, it implicitly learns to account the limitation over time based on how the pursuer executes given actions.
deep	"Deep" learning approaches use multiple layers in an artificial neural network. In our network, we have two "hidden" layers between the input layer (state observations) and the output layer (heading & acceleration).
deterministic	"Deterministic" methods do not incorporate any randomness into their policies. That is, for the same inputs (state observations), you get the same outputs (heading & acceleration actions).
policy gradient	"Policy gradient" methods move the policy (set of heading & acceleration actions) in the direction of the gradient of improvement (higher pursuit rewards), in the context of DDPG with respect to the critic model reward expectation.

The resulting RL policy was tested under many scenarios and performance exceeded that of a baseline bearing-following strategy (Section 4.1). Emerging RL success strategies were analyzed and manually abstracted into grouped behaviors. Finally, an online decision-making framework is discussed for leveraging conditional past performance to predict future performance, as explored further in Chapter 3 & Chapter 4.

Section 1 Introduction

Chapter 1 introduced potential real-world applications, an overview of the pursuit-evasion game, a summary of previous research related to pursuit games, and novel areas of research contributed by this chapter. Section 2 of this chapter describes the simulation environment of the pursuit and target agents, the game environment and associated OpenAI Gym implementation, and uncertainty modeling assumptions. Section 3 outlines an introduction to RL and the implemented continuous deep RL approach. Section 4 presents results and discusses their comparison to a baseline alternative algorithm. And Section 5 summarizes conclusions and future work.

Section 2 Simulation Overview

In this chapter we consider a two-dimensional pursuit game with variable-target speed and variable-pursuer speed. There are many 2-d applications in the real world, such as those associated with unmanned ground and maritime surface robotics, and others discussed in Chapter 1. However, insights gained from analyzing a 2-d game could also have relevance to a 3-d extensions. The initial position and heading of both the target and pursuer are randomized uniformly over 30,000 simulated games for training and 5,000 simulated games for testing to make the computational results more general. The target holds a near-constant heading and speed, allowing for slight maneuvers along its trajectory modeled as process noise and sampled from a multi-variate Gaussian distribution with a mean of zero and a standard deviation of 1e-10 m^2/s^3, applied to velocity and acceleration vector components proportional to time in accordance with the process noise matrix (Q) as given in Table 5. The pursuer then attempts to capture the target by maneuvering within a distance smaller than the capture radius (ε). The environmental and Kalman filtering parameters in Table 5 were used to perform the experiments described in Chapters 2 & 3.

Table 5 Environmental Run Parameters

Parameter	**Value**
simulation time step (dt)	1 second (s)
maximum simulation duration	500 s
heading change rate limitation	30 degrees per second
maximum acceleration	4 (m/s^2)
maximum pursuer speed	2 m/s
capture radius (ε)	50 m
maximum distance from target	1200 m
simulated sensor standard deviation error for angle measurements	3 degrees
simulated sensor noise standard deviation for range measurements	50 meters
transition matrix (F)	$\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
process noise matrix (Q)	$\begin{bmatrix} dt^3/3 & 0 & dt^2/2 & 0 \\ 0 & dt^3/3 & 0 & dt^2/2 \\ dt^2/2 & 0 & dt & 0 \\ 0 & dt^2/2 & 0 & dt \end{bmatrix}$
GPS position error standard deviation	10 m
GPS velocity error standard deviation	0.1 m/s

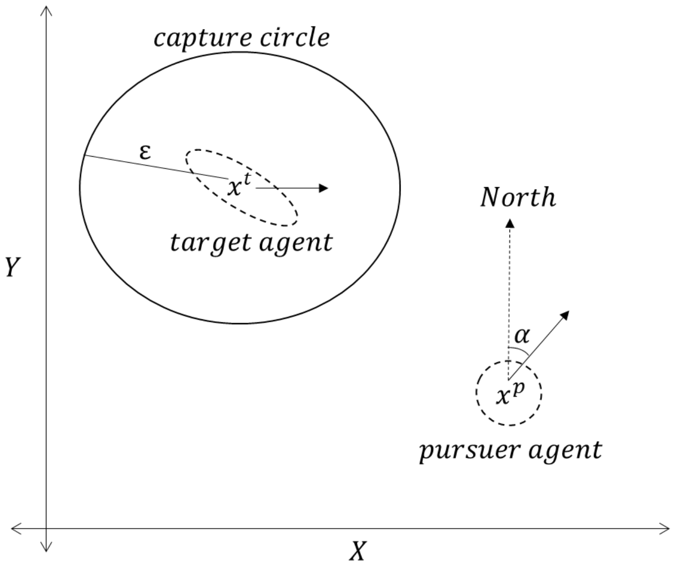


Figure 1 Pursuit game illustration

In Figure 1, we show a map in space where the X axis represents East-West orientation and Y represents North-South orientation. The target position and associated uncertainty are represented as the dashed ellipse depicted around x^t while the pursuer is represented by the dashed circle around the state represented by x^p.

For the target, we simulate measurements obtained from the pursuer's position, so the elliptical uncertainty correlates with the angle in which the measurement was taken. For the pursuer, we assumed a fixed uncertainty covariance matrix at the equivalent of a conservatively accurate global positioning system (GPS) measurement (10 meters), uniform in all directions, yielding a circle.

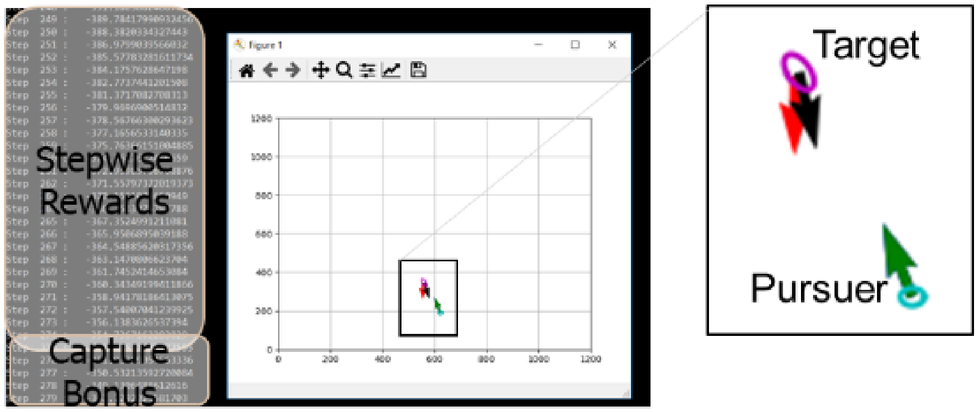
A four-dimensional state vector is defined for both pursuer (x^p) and target (x^t) agents. The first two state vector components represent the agent's position (in 2-d) and the second two components represent the agent's 2-dimensional velocity. Note that only the position components of the state estimate and their uncertainties are illustrated in Figure 1. Each agent's state is an imperfect, mean estimate on its position and velocity. In this research, we consider pursuer and target states with uncertainty and we model uncertainty using a 4-by-4 covariance matrix P for both the pursuer (P^p) and the target (P^t) which represent the accuracy of the mean state estimate. Uncertainty estimates are propagated forward in time and updated when new measurements are received using a Kalman Filtering approach, Equations 2 & 6, respectively, discussed in Section 2.1.

The solid arrows emanating from each agent in Figure 1 represent the agents' respective direction of travel and the dotted arrow depicts a "North"-orienting direction. The angle α represents the heading of the pursuer. The circle about the target position represents the capture radius; when the pursuer enters within the radius ε, it "captures" the target and wins the game. To represent games with more stringent or liberal capture requirements, one can reduce or lengthen the capture radius.

Lastly, in this game implementation, the target does not attempt to evade the pursuer. Thus this is a pursuit-only game as studied in [12, 13], where the pursuer and target agents are present in the game, but only the pursuer is being controlled. In this pursuit-only game, the target agent knows nothing of the pursuer or that it is being pursued. This is relevant for applications where the target agent is not equipped with intelligent capabilities or where the pursuer is attempting to sneak up on the target. The pursuer, however, knows an estimate of the target's position & velocity state, and the associated uncertainty of its state estimate.

This approach could be scaled to multiple target agents and pursuer agents by expanding the dimensions of the action and observation spaces, accordingly. Additionally, the Kalman filter-targeting approach would need to be extended to cover multiple targets. However, in the case where the number of targets is unknown, a multi-hypothesis tracking approach [31] should be used in place of a Kalman filter and the deep RL observation space would need to be set to some maximum number of targets (which can become intractable and is very inefficient); future research should consider support for flexible observation-sized inputs into learning approaches, an open research area.

Additionally, this approach could be extended to two players playing a pursuit-evasion game, where the target agent becomes an evader agent. For this extension, two instances of DDPG algorithms are trained simultaneously over the same environment; preliminary results support that the approach extends to the pursuit-evasion game without issue, however, these results are omitted from this dissertation. We also note that any extensions of the game can take advantage of the previously trained network in order to bootstrap learning of a modified game via transfer learning. Transfer learning approaches take advantage of machine skills learned (encoded into network weights) for a previous task to inform the learning of a new task. In other words, by using initial network weights that correspond to a previously-learned task, we accelerate the learning of a new, similar task.



```
Step 249 :   -389.78417990032456
Step 250 :   -388.3820334327443
Step 251 :   -386.9799033566032
Step 252 :   -385.57783281611734
Step 253 :   -384.1757628647198
Step 254 :   -382.7737441201508
Step 255 :   -381.3717082708313
Step 256 :   -379.9696900514832
Step 257 :   -378.56766300293623
Step 258 :   -377.1656533140335
Step 259 :   -375.7636515100488
[illegible]
Step 262 :   -371.55797322019373
[illegible]
Step 265 :   -367.3524991211081
Step 266 :   -365.9506895039188
Step 267 :   -364.54885620317356
Step 268 :   -363.1470006623704
Step 269 :   -361.7452414653084
Step 270 :   -360.34349199411866
Step 271 :   -358.94178186413075
Step 272 :   -357.54007041239925
Step 273 :   -356.1383626537394
[illegible]
Step 277 :   -350.53213592720084
Step 278 :   [illegible]
Step 279 :   [illegible]
```

Figure 2 Python Pursuit Game Rendering

In Figure 2, we share the Python interface we developed that shows the pursuer and target agents in a 2-d map. Here, we display both the true target position & heading (black vector) and estimated target position & heading (magenta ellipse and red vector, respectively) consistent with the underlying observations. Similarly, the pursuer true position and heading are displayed (green vector), along with its associated position uncertainty (light blue ellipse). In this example, the pursuer estimated heading was aligned with the truth, so the estimated heading vector is not displayed. In the figure (left), we show the outputs from each step through the game episode and their associated rewards, discussed in detail in Section 2.5. This developer interface was useful for debugging the approaches and understanding the agent's behavior under different environmental conditions (scenarios).

2.1 Measurement Simulation

Unlike previous research (Table 1), the pursuit game studied in this chapter incorporates uncertainty on both the pursuer and target positions consistent with real-world expectations. The pursuer only senses the approximate target position, resulting in imperfect information. More specifically, the pursuer perceives a line of bearing measurement to the target (angle) and distance measurement to the target (range) at each simulated time step. A measurement model was implemented which accounts for pursuer-to-target relative geometries and randomized measurement error. A Kalman filter dynamical system model [32] was chosen to predict and track the mean state estimate and uncertainty over time according to the dynamics in Equations 1-6 (see below). Due to Kalman filtering's memoryless property, only the current measurement and the state estimates from the previous time step are needed to calculate an updated target estimate; no history of measurements needs to be stored. The Kalman filter is an optimal measurement-fusion technique used to estimate states based on linear dynamical systems in state-space format. Pairing the Kalman filtering approach with a RL approach was natural, as both use state estimates to describe the environment. In this case, the states estimated by the Kalman filter were directly included into the observation space (state) used by the RL approach. Moreover, Kalman filtering has been used for position tracking applications for decades, initially applied in the 1960's to space trajectory estimation and integrated on the Apollo computer [33].

A Kalman filter consists of two overarching steps: the prediction step and the update step. The prediction step (Eq. 1-2) propagates the state estimate forward in time and the update step (Eq. 3-6) revises the predicted state estimate with the information gained from incoming measurements. For example, if the pursuer senses the target at some time and wants to estimate the target's position at a future time, it must propagate the position, velocity, and associated error covariance estimates over time in accordance with its velocity estimate. Otherwise, the pursuer would not anticipate where the target would be with accuracy. This state propagation is achieved through the prediction step (Eq. 1-2).

$$Predicted\ state\ estimate\colon\ x_k = Fx_{k-1} + w_{k-1} \qquad (Eq. 1)$$

$$Predicted\ error\ matrix\colon\ P_k = FP_{k-1}F^T + Q \qquad (Eq. 2)$$

$$with\ w_{k-1} \sim N(0, Q)$$

where k is the current simulation step, x_k is the 4-d target state vector (2-d position and 2-d velocity), F is the state transition matrix, and w is the process noise vector sampled from a Gaussian distribution with zero mean and covariance (process noise) Q. Moreover, when a new measurement is received, the state estimates need to be updated in accordance with the information provided. This is achieved by the Kalman update steps (Eq. 3-6):

$$Measurement\ residual\colon\ \tilde{y}_k = z_k - Hx_k \qquad (Eq. 3)$$

$$Kalman\ gain\colon\ K_k = P_kH^T(R + HP_kH^T)^{-1} \qquad \text{(Eq. 4)}$$

$$Updtated\ state\ estimate\colon\ x_k = x_k + K_k\tilde{y}_k \qquad (Eq. 5)$$

$$Updated\ error\ covariance\colon\ P_k = (I - K_kH)P_k \qquad \text{(Eq. 6)}$$

where z is the target position and velocity measurement vector, R is the measurement noise matrix, and H is the measurement matrix [32]. Now that we have shared how to propagate estimates forward in time via the predict steps (Eq. 1-2) and how to update the estimates based on an incoming measurement via the update steps (Eq. 3-6), we share how we incorporate our state estimates into our RL observation space.

2.2 Observation Space

With the natural compression of uncertainty into covariance matrices via Kalman filtering and the symmetry of covariance matrices, the observation space is compactly described as 18 elements partitioned here for explanation into three categories:

1) Elements 1-4 are the target mean state estimate elements

2) Elements 5-14 are the 10 lower triangular unique entries in the target covariance matrix, collapsing the covariance matrix into a single dimensional vector

3) Elements 15-18 are the pursuer mean state estimate elements

The target agent's covariance matrix can be reduced to 10 unique elements since the matrix is symmetric. Here is where we incorporate the learning of uncertainty into the RL approach, a novel element of this study. At each time step, this 18-element observation is formed by executing the Kalman update from simulated measurements of sensors native to the pursuer agent. This observation allows the RL pursuit agent to interpret the target position and velocities, their associated uncertainties, and the estimated pursuer agent position before choosing an action. Depending on the action chosen, new measurements is simulated, and the observation will be updated accordingly.

2.3 Action Space

The action space is comprised of two elements: acceleration (a) and change in heading (φ). In order to enable velocity control, we include an acceleration decision variable (in meters per second squared (m/s^2)). The inclusion of an acceleration decision variable is another novel element of this study, addressing dynamic speed control (Table 1). Additionally, a change in heading (in radians) decision variable is included. Most literature in homicidal chauffeur and pursuit-evasion games surveyed in [6] [7] only include the change in heading (one-dimensional) in the pursuer and evasion decision spaces.

2.4 Game Progression

At each simulation step, the change in heading action is applied by taking the rotation matrix formed by the chosen angle change (φ) and applying it to the pursuer velocity, as shown in Eq. 7, where the third and fourth velocity components of the pursuer's 4-d state estimate vector are annotated with their associated subscripts and the updated velocities are annotated with a prime designator ($'$). Additionally, the acceleration action is applied by increasing or decreasing the pursuer velocity along the new heading, as shown in Eq. 7 where dt is the simulation time step and a is the acceleration action.

$$\begin{bmatrix} x_3^{p\prime} \\ x_4^{p\prime} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} x_3^{p} \\ x_4^{p} \end{bmatrix} + dt \begin{bmatrix} a \\ a \end{bmatrix} \qquad (Eq.7)$$

A new measurement, range and azimuth to target, is generated by accounting for the relative geometries between the pursuer (equipped with a range and azimuth sensor) and the target. First, the true range and azimuths are computed. Second, noise is added according to a Gaussian distribution with zero mean and predefined range and azimuth noise levels (Table 5, simulated sensor noise standard deviation for range measurements & simulated sensor standard deviation error for angle measurements).

2.5 Reward

The reward applied at each step is the calculated distance between the pursuer and the target (Eq. 8), using the Euclidean norm. Since RL algorithms are most commonly designed to maximize reward, the resulting distance is inverted in sign. This encourages the pursuer to get closer to the target at each step. The simulation time is a parameter, along with others provided in Table 5; here, we used a simulation time step of 1 second.

$$r = -\left\|\left(x_t - x_p\right)\right\| \qquad (Eq. 8)$$

Three different terminal (sparse) rewards were applied using the following scenarios.

1) If the pursuer's distance to the target exceeded the maximum threshold (1200 meters), a penalty of -1500 was applied.
2) If the maximum simulation step limit (500 seconds) was reached, a penalty of -1500 was applied.
3) If the pursuer penetrated within the capture radius of the target, a reward of 1500 was applied minus the total pursuit time duration (from scenario start to time of capture) to encourage more efficient pursuit strategies.

The selection of the penalty in 1) was chosen so that it exceeded the possible distance from the target reward at any time step (-1200). Otherwise, the agent could be rewarded for traversing further away from the target. The selection of the capture reward was chosen so in an immediate capture scenario, the reward would be symmetric to the failure rewards in 1) and 2) about zero. Since the maximum time is limited to 500 time steps, the lowest capture reward the pursuer could earn was 1000 (1500 – 500 time steps). In all three of these scenarios, the game ends. No other termination criteria were specified.

Section 3 Reinforcement Learning Approach

3.1. RL

RL is a subset of ML wherein an agent learns by interacting with an environment through its actions and their associated rewards and effects, exactly like a Markov Decision Process (MDP), but where all of the states and transitions are not necessarily known. The goal of a MDP or an RL approach is to determine an optimal control policy (sequence of actions) which maximizes all future rewards. The basic agent-environment interaction modeled in RL is illustrated in Figure 3.

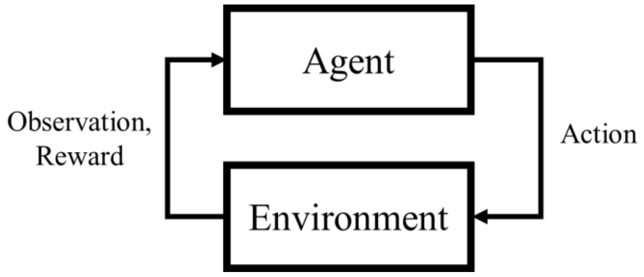


Figure 3 RL approaches learn optimal policies to maximize rewards in a given observable environment

Deep reinforcement learning is particularly well-suited to handle sparse rewards like those experienced in this pursuit game, as ratified by the breakthrough AlphaGo performance in [34]. Due to the continuous nature of the action and observation spaces in this pursuit game, and the success of its employment in [11], the DDPG algorithm in [30] was used. The DDPG approach combines the Deterministic Policy Gradient (DPG) reinforcement learning algorithm from [35] with deep learning function approximation. An alternative asynchronous actor-critic (A3C) RL approach was explored by the authors for this exact problem implementation, but the trained agent did not achieve satisfactory performance even after long periods of training on near-stationary targets; the authors concluded that the A3C approach attempted was not successful due to the attempt to discretize the intractable continuous action space inherent to the pursuit game.

3.2 DDPG

The DDPG algorithm in [30] combines actor and critic methods with policy-gradient methods. Actor-critic methods consist of two models; the actor model which gives an action for a given state and a critic model which anticipates the reward of a given action based on a given state. The actor model selects an action which maximizes an approximate Q-function which the critic learns by minimizing a Bellman loss function.

A Bellman equation first applied to dynamic programing, where a problem is decomposed into a sequence of subproblems, determines an optimal policy by recursively solving an action-value function $Q^{\mu}(s_t, a_t)$ (Eq. 9), where s_t and a_t are the state and action at time t, respectively, r is thre reward function, E is the environment described in Section 2, μ is the target policy, and γ is the discount factor applied to future rewards.

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))] \ (Eq. 9)$$

DDPG employs two sets of actor-critic agents, each rooted in a deep artificial neural network. In addition to actor-critic networks updated at every time step, an additional set of actor-critic target networks are updated at a slower rate, and only with some of the updated weights from the current actor-critic networks, making for smoother model updates, less sensitive to uncommon scenario-specific interactions and edge cases. The DDPG algorithm from [30] is described in Table 6 in detail for this implementation. Essentially, the algorithm aims to optimize the weights of two networks. The actor learns network weight parameters that produce “good” pursuer heading & acceleration actions based on state observation inputs. The critic learns network weight parameters that estimate the pursuit reward associated with state and action inputs.

Table 6 DDPG Algorithm applied to the Pursuit Game

```
Algorithm 1 DDPG Algorithm applied to the Pursuit Game
Parameters: Discount factor $\gamma$, number of episodes $M$, number of steps for each episode $T$, batch size $n$
1. Randomly initialize critic network $Q(s, a | \theta^{Q})$ and actor network $\mu(s|\theta^{\mu})$ with weights $\theta^{Q}$ and $\theta^{\mu}$.
2. Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$
3. Initialize replay buffer $R$
4. for episode = 1, $M$ do
   5. Initialize a random process $N$ for action exploration
   6. Receive initial observation state $s_1$, where the state is comprised of the 18 elements described in Section 2.2.
   7. for t=1, T do
      8. Select action $a_t = \mu(s|\theta^{\mu}) + N_t$ according to the current policy and exploration noise, where the action is comprised of the 2 elements described in Section 2.3.
      9. Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$, where the pursuer reward is described in Section 2.5.
      10. Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
      11. Sample a random batch of $n$ transitions $(s_t, a_t, r_t, s_{t+1})$ from $R$
      12. Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      13. Update critic by minimizing the loss using a gradient descent optimizer:
              $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^{Q}))^2$
      14. Update the actor policy using the sampled policy gradient:
              $\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a | \theta^{Q})|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s_i}$
      15. Update the target networks:
              $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$
              $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
   end for
end for
```

The terms used in the DDPG algorithm outlined above are summarized in Table 7.

Table 7 DDPG Terminology

DDPG Term	**Short description**
γ	Discount factor applied to future rewards.
$\boldsymbol{M}$	Number of episodes. Number of pursuit games to play during the DDPG training phase. For us, this is 30,000 pursuit games.
$\boldsymbol{T}$	Number of steps for each episode. This is maxed out at 500 steps for our pursuit games.
$\boldsymbol{t}$	Current time step index.
$\boldsymbol{n}$	Batch size. Governs the number of transitions that are sampled from the replay buffer to compute the expected game reward.
θ^Q	Weights in the current critic network.
θ^μ	Weights in the current actor network.
Q	Current critic neural network. The neural network that governs reward function approximation.
μ	Current actor neural network. The neural network that governs action selection.
Q'	Target critic neural network.
μ'	Target actor neural network.
$\theta^{Q'}$	Weights in the target critic neural network, updated at a slower rate than the current critic neural network.
$\theta^{\mu'}$	Weights in the target actor neural network, updated at a slower rate than the current actor neural network.
R	The replay buffer. Storage of transitions saved in memory for later reference. Here, we save transitions from a current state to the next state, in accordance with our acceleration and heading actions, and the associated reward.
N_t	Exploration noise for time *t*. Noise is added to the action itself to promote exploration of the state space.
a_t	Action for time *t*, where the action is composed of two parts, heading and acceleration, as described in Section 2.3.
s_t	State for time *t*, where state is composed of 18 elements described in the observation Space Section 2.2.
r_t	Reward for time t, where the pursuer reward is described in Section 2.5.
L	Loss function. Bellman equation as described in Section 3.2.
$\nabla_{\theta^\mu} J$	Sampled policy gradient. Take the mean of the sum of gradients calculated from the mini-batch experiences in the batch indexed by *i*.
$\boldsymbol{\tau}$	Proportion of target network weights that are updated from the current networks.

Specific DDPG implementation parameters and their values are listed in Table 8.

Table 8 DDPG Run Parameters

Parameter	**Value**
optimizer	Adam [36]
target network update parameter (τ)	0.99
batch size (*n*)	64
actor and critic linear neural net structures	18 (inputs/Section 2.2) x 400 (hidden layer) x 300 (hidden layer) x 2 (outputs/Section 2.3)
discount factor (γ)	0.99

Section 4 Results & Discussion

After execution of 5000 randomized testing trials, we compared the DDPG-trained agent capture performance to that of a baseline bearing-following (always taking a direct angle to the target) pursuit approach. Moreover, we studied performance sensitivities to initial conditions. Each trial consisted of initial position randomization of the pursuer and target within a 1200 by 1200 meter (m) operating space and randomized maximum target speed between 0 and 8 meters per second (m/s). Sensitivities to performance were analyzed for three initial conditions:

1) The starting distance of the pursuer from the target
2) The initial relative angle to the target
3) The maximum speed of the target

4.1. Bearing-Following Baseline Algorithm

The baseline bearing-following capture strategy takes a direct route to the target from the pursuer's position using the algorithm described in Table 9, and further illustrated in Figure 4.

Table 9 Baseline bearing-following pursuit algorithm

```
Algorithm 2 Baseline bearing-following pursuit algorithm
1. Compute the pursuer to target position bearing vector $\boldsymbol{v}$
2. Compute the angle from the pursuer to the target, where $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are the Cartesian X and Y components of the vector $\boldsymbol{v}$
       $\boldsymbol{\theta} = \textbf{arctan}\left(\frac{\boldsymbol{v_2}}{\boldsymbol{v_1}}\right)$
3. Compute the current pursuer heading $\boldsymbol{\alpha}$
4. Compute the maximum heading action which will align the pursuer heading with the velocity heading
       $\boldsymbol{\varphi} = \boldsymbol{\omega} * (\boldsymbol{\theta} - \boldsymbol{\alpha})$
5. Continue at constant speed (zero acceleration) and apply change in heading action $\boldsymbol{\varphi}$
```

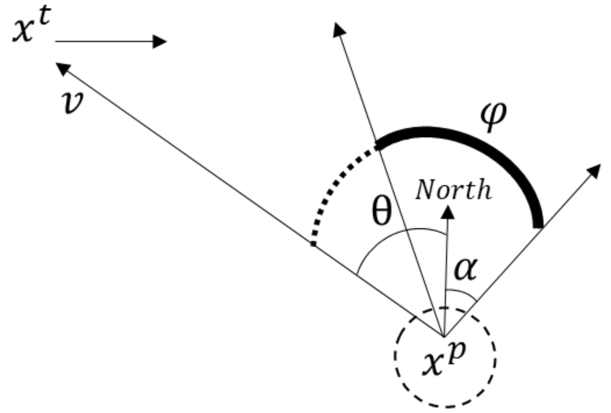


Figure 4 Baseline bearing-following policy illustration

The bolded action $\boldsymbol{\varphi}$ complies with the $\boldsymbol{\omega}$ limit on maximum heading change, analogous to the turning radius limitations studied in the homicidal chauffeur problem. In our case, $\boldsymbol{\omega}$ was equal to 30 degrees per second, governing how fast the agent can turn. That is, the agent can only turn so much in one simulation time step. In the baseline pursuit strategy, the pursuer maintains a constant speed, consistent with previous literature [12, 14, 9, 11] and the classic homicidal chauffeur problem [5] and its derivatives surveyed in [6, 7].

4.2. Baseline-RL Results Comparison

The DDPG agent was trained using a Pytorch platform on an HP ZBook 15 computer with a modest 16GB of memory and an Intel Core i7-6820HQ CPU. A training episode is analogous to running the pursuit game simulation once. The DDPG agent analyzed in this Chapter was trained against 30,000 training episodes; 2,000 training episodes were timed and took approximately 3 hours of computational runtime without any efforts for parallelization. The timing scaled nearly linearly for 30,000 episodes, taking approximately two days of processing time, though the experiment was not explicitly timed.

The following figures show the performance of the RL agent compared to the baseline agent. The X axis represents the discretized bins associated with the corresponding initial conditions, using the lower limit bin value as the axis label. Figure 5 shows the performance superiority of the RL agent over the baseline agent with respect to the initial speed of the target. The X axis represents the maximum speed of the target and the Y axis represents the number of cumulative (aggregated) captures as the maximum speed increases. Here, based on superior performance, we know that the RL agent has learned anticipatory strategies sufficient to plan an interception point in advance of the target's current position. Otherwise, its performance would present the same or worse than the baseline algorithm. In the cases where the target maximum speed is greater than that of the pursuer's (where the X axis value is greater than 2 m/s), the pursuer is successful in a speed-overmatched game. This result is novel and no prior work has realized a RL pursuer agent capable of pursuing a speed-advantaged target.

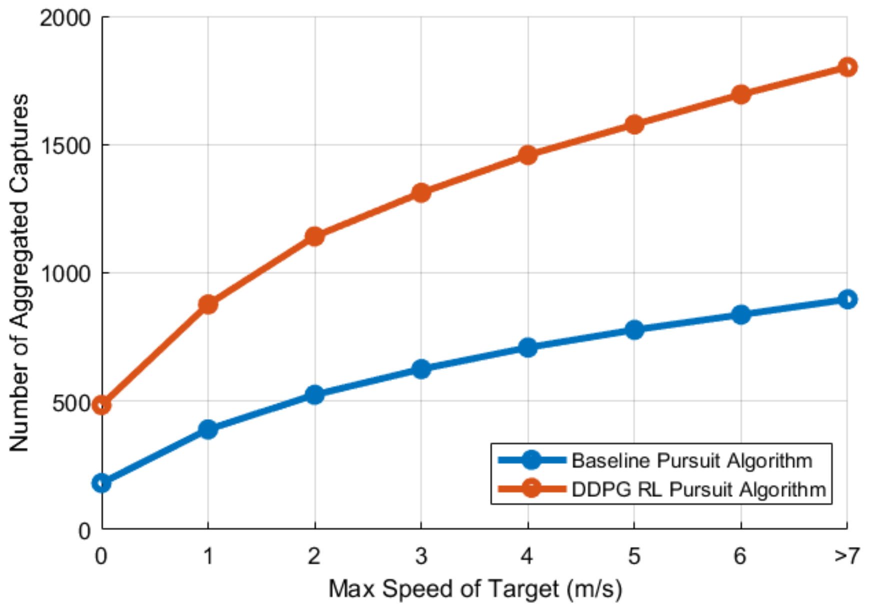


Figure 5 DDPG RL pursuit algorithm outperforms baseline algorithm against increasing target maximum speed parameters

The non-cumulative captures as a function of maximum target speed, as shown in Figure 6.

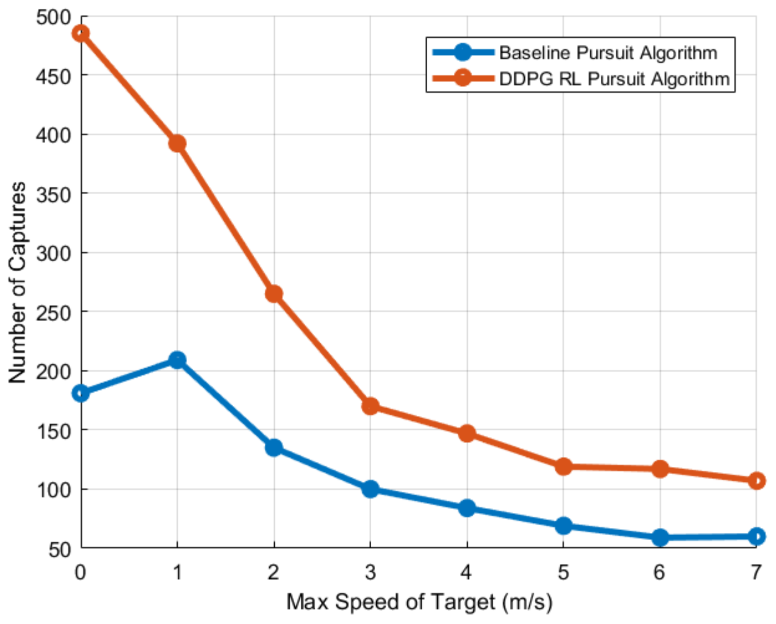


Figure 6 DDPG RL pursuit algorithm outperforms baseline algorithm in aggregated captures against initial angle to target conditions

Figure 7 shows the performance sensitivities of both algorithms to the initial distance between the pursuer and the target.

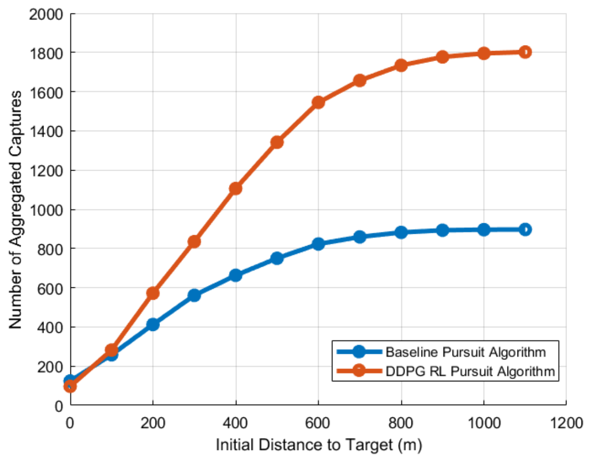


Figure 7 DDPG RL pursuit algorithm outperforms baseline algorithm against increasing initial distance to target parameters

The non-cumulative plot of the initial distance to target effects on baseline and RL-trained performance is shown in Figure 8. Here, the performance advantage of the DDPG algorithm over the baseline algorithm becomes exaggerated early in the parameterization space, achieving significant separation around 200 meters, but then shrinks around 900 meters.

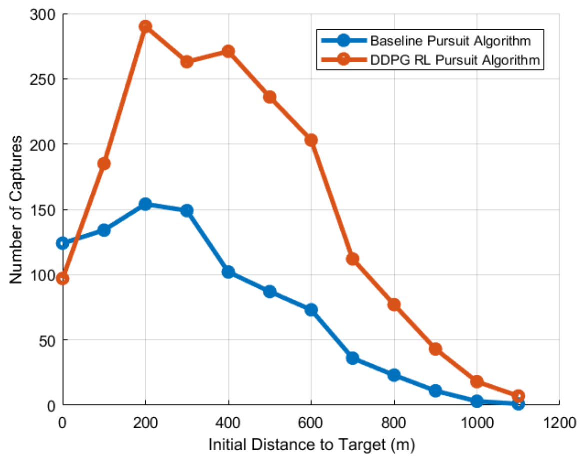


Figure 8 DDPG RL pursuit algorithm outperforms baseline algorithm in aggregated captures against initial angle to target conditions

Figure 9 shows the non-cumulative captures as a function of the initial angle of the pursuer to the target. This plot illustrates an expected result in which the baseline algorithm performs at a level commensurate with the RL algorithm when positioned "ahead" or "in front" of the target (orange rectangle). Here, we would expect a simple bearing-following (direct angle to the target position) strategy to be successful, and it is.

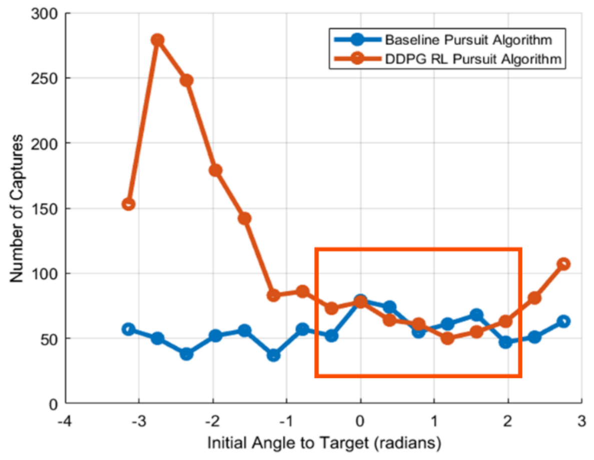


Figure 9 Baseline pursuit algorithm performs well in certain pursuer-to-target angle initializations

The cumulative (aggregated) plot of the initial angle effects on baseline and RL-trained performance is shown in Figure 10.

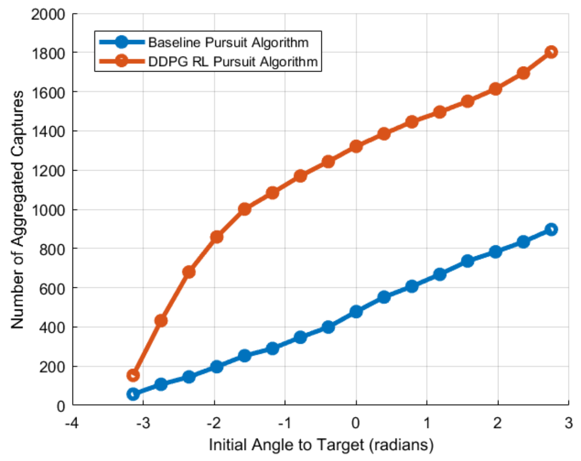


Figure 10 DDPG RL pursuit algorithm outperforms baseline algorithm in aggregated captures against initial angle to target conditions

4.3. Comparison to DDPG Approach without Uncertainty

In addition to analyzing the effects of the RL-based solution against a course-aligned baseline control policy, we also performed an experiment to quantify the benefits of the inclusion of the covariance error into the observation space. Here, the new baseline algorithm consists of the same DDPG approach with a modified observation space. While the original observation space contained 18 elements, including the error covariance matrix for the target, the modified observation space only contains 4 elements, corresponding to the mean 2-d position estimate for the pursuer and the target.

When we train each algorithm over 10,000 trials and perform an experiment over 5,000 trials, we find that the RL algorithm that directly accounts for uncertainty in its observation space outperforms the baseline RL approach without uncertainty by a factor of 4 times. The RL algorithm with uncertainty wins 1011 of 5000 games while the baseline algorithm only wins 252 of 5000 games. The fraction of captures over the initial distance to target condition is shown in Figure 11.

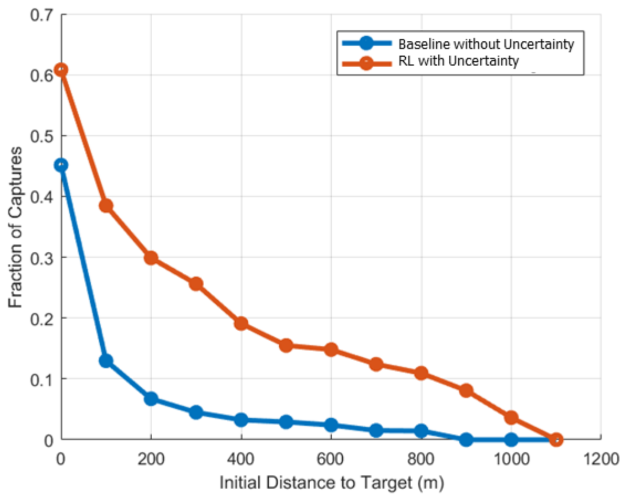


Figure 11 Comparison of RL pursuit algorithm with uncertainty to baseline RL pursuit algorithm without uncertainty

This result shows the importance of encoding known information about uncertainty directly into the observation space. However, the RL algorithm may be able to learn the uncertainty on its own over time based on uncovering the relationship between the relevant pursuer-to-target geometries, consistent with simulated measurements. In any case, the incorporation of an error covariance tracker for pursuit games with uncertainty shows considerable benefits over its exclusion.

4.4. Online Decision-making Leveraging Conditional Past Performance

While randomness was controlled between RL and baseline experiments, the sampling from the initial parameters was not deliberately controlled for the initial angle to the target and distance to the target since each trial was initialized by placing the pursuer and target agents down randomly in the operating space. Thus, the number of samples across the initial distance to target and initial angle to target parameter spaces were not evenly sampled.

Regardless, in the next set of results, we consider the historical performance of the DDPG RL algorithm for supporting the decision of whether to pursue an agent given the current conditions. Figure 12 shows the experimental capture success, as a fraction of wins over total trials, in a heat map over the coupled conditional effects of the initial distance to the target and the maximum speed of the target.

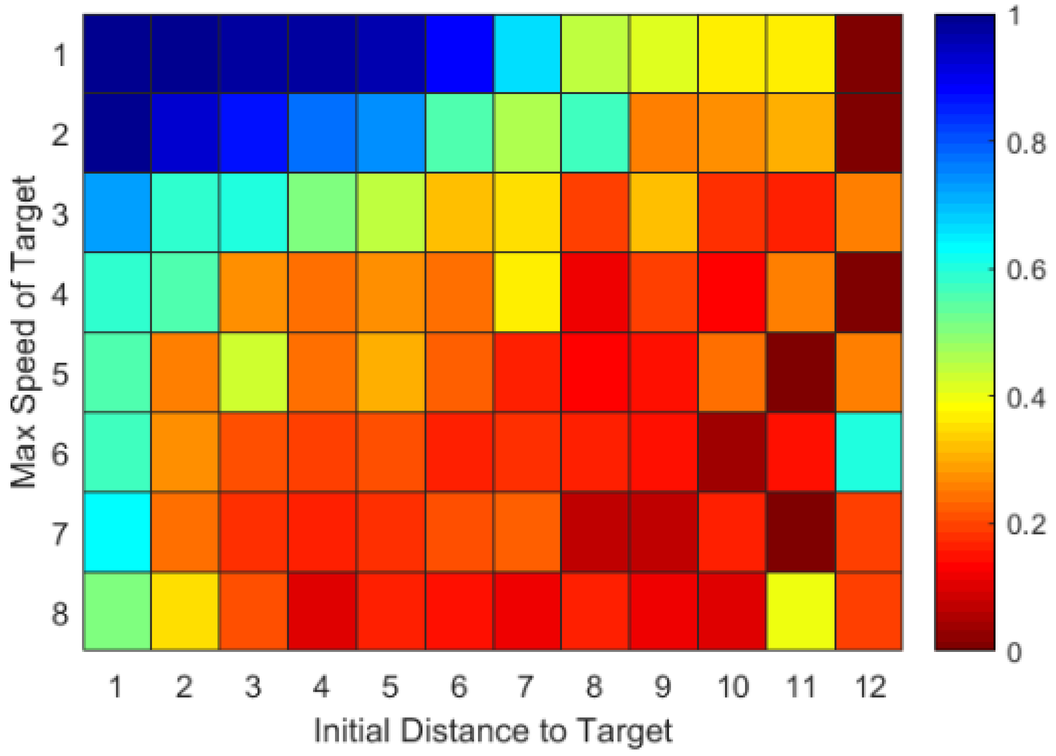


Figure 12 Heat map of DDPG RL performance under coupled conditions for 5000 sample games

Further analysis of Figure 12 indicates that when the pursuer starts close to a slow-moving target (upper left of heat map), its performance supports high capture success. Conversely, as the pursuer's initial distance to the target grows and the maximum speed of the target increases, the chances of successful capture decrease.

We note the anomalous historical capture success cells in the 11th and 12th distance bins and explain them by recalling that the number of samples across the horizontal bins was not uniform. Only 5 samples were analyzed for each of those two anomalous parameter combinations resulting in 2 of 5 and 3 of 5 successful captures for the 11th and 12th column anomalous cells, respectively.

4.5. RL Agent Strategy Abstraction

Our final analysis segment focuses on categorization of the strategies which emerged out of the DDPG RL optimization policy. These motivate the automated strategy mechanism employed in Chapters 3 & 4. Figure 13 shows the paths of the first 17 successful runs of the 5000 test cases and human visual inspection supports that the pursuer follows five different types of strategies:

1) “L-shaped”: The pursuer closes in on the target perpendicular to the target’s direction of travel; then, the pursuer turns approximately 90 degrees toward the target and closes distance until capture.
2) “Direct”: The pursuer takes an efficient route toward the target, anticipating the capture point near-perfectly.
3) “Unsure”: The pursuer attempts to improvise a favorable approach angle to the target, unsure of the capture point.
4) “Race”: The pursuer runs near-parallel to the target until it catches up, then turns toward the target to secure a capture.
5) “Sweep”: The pursuer attempts to approach the target along an arc, leading to a successful capture.

Interestingly, these strategies are determined automatically through the introspective methods in Chapter 3 (Figure 20). In Figure 13, we manually labeled the scenarios with each of the five strategy abstractions discussed here.

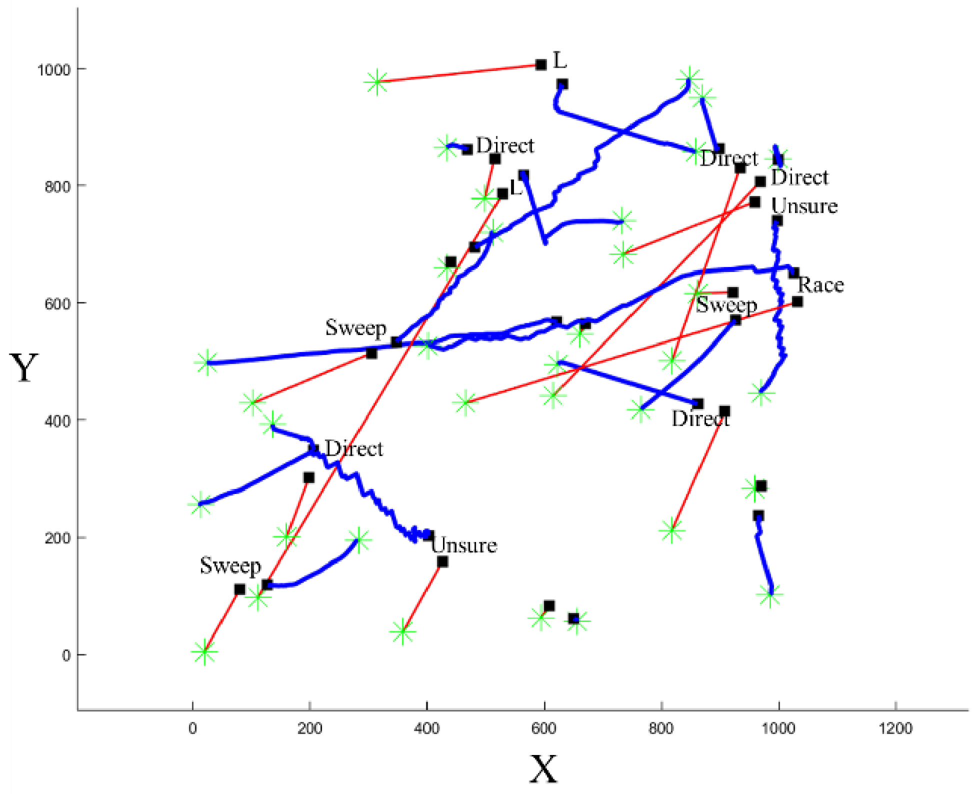


Figure 13 17 successful run samples and their associated manual strategy abstraction labels in X-Y space

Successful captures without labels did not have discernable qualities or were trivial scenarios where the target moved especially slow.

Section 5 Conclusions

We demonstrated the utility of an agent trained with the DDPG reinforcement-learning algorithm in a speed-overmatched pursuit game with uncertain target information. The RL agent outperformed a baseline bearing-following strategy by increasing capture successes by more than 100% in a 5000-trial experiment. Furthermore, the RL agent was more robust to harsher distance and angle-to-target starting conditions and overmatched target speeds. As the target speed increased, the benefits of the RL approach over the bearing-following baseline strategy widened greatly. Future research could explore the introduction of obstacles and their effects on agent performance, as explored in [10] via mixed complementarity problem modeling.

Additionally, we discussed the potential utility of leveraging historical performance partitioned by initial conditions for online unmanned system decision-making. Future work could consider integrating conditional historical performance data into an online capability which predicts the probability of successful capture from historical data and current operating conditions. This capability could help the pursuer determine whether it should pursue the target of opportunity, wait for a different target, or avoid resource expenditure under unlikely success situations; examples include expending energy to attempt to intercept a pass in sports or expending limited ballistic missile defense resources toward incoming threats. Such analysis would also help determine the timing for when the pursuer should begin pursuit, leading to better energy efficiencies for unmanned systems with endurance limitations.

Moreover, we analyzed the underlying strategies employed by the RL-trained pursuer agent against the target agent. We manually inspected the resulting paths and aggregated them into five labeled categories. Future work could investigate online strategy and inference, such as methods for online strategy and performance prediction, prototyped in Chapter 4.

Chapter 3: Uncovering Strategies and Commitment through Machine Learning System Introspection

Section 0: Overview

Deep neural networks are naturally "black boxes", offering little insight into how or why they make decisions. These limitations diminish the adoption likelihood of such systems for important tasks and as trusted teammates. We employ introspective techniques to abstract machine activation patterns into human-interpretable strategies and identify relationships between environmental conditions (why), strategies (how), and performance (result) on a deep reinforcement learning 2-d pursuit game application. "Introspection" here refers to the analysis of neural network activation patterns, and is analogous to looking into one's own brain from psychology. Activation patterns refer to the outputs of each neural network hidden layer over time and are considered synonymous with machine "behaviors". For example, we found that activation patterns that were abstracted into "head-on" or "L-shaped" maneuver strategies were successful and intuitively corresponded to favorable initial conditions, such as the initial distance to the target and the maximum speed of the target. In this time-series application, we are performing introspective analysis after the game has been played. As we have shown considerable utility in studying machine strategies, we motivate future research into development of an online strategy *predictor*. An online strategy predictor could provide near real-time updates to a human partner about the estimated strategy that the machine is using to perform the task. As a result, the human can anticipate the strategy that the ML is employing prior to making important actions, like a 90-degree turn associated with an "L-shaped" maneuver strategy.

We are interested in characterizing machine strategies abstractly so that humans can gain insights into the "black box", understanding how the machine arrived at its output rather than accepting it blindly. For example, a human is more likely to accept an ML output of an "airplane" classification of an object in an image if the machine says it used the "looked for wings" strategy rather than just accepting the result without any understanding of how the machine determined that an airplane was present.

Moreover, we characterize machine commitment by the introduction of a novel measure and reveal significant correlations between machine commitment, strategies, environmental conditions, and task performance. By uncovering temporally dependent machine "thought processes" and commitment through introspection, we contribute to the larger explainable artificial intelligence initiative, increasing transparency and trust in machine learning systems. And we motivate online exploitation of machine behavior estimation for competency-aware intelligent systems by revealing correlations between strategies, commitment and resulting *performance*.

The motivation and summary of prior work (n Table 2.

Table 2) pertinent to this research area are discussed in Chapter 1. In this chapter, we uncover and analyze machine behaviors through dimension reduction and time-series clustering for a RL agent playing a 2-dimensional pursuit game, as introduced in Chapter 2. In the game, a pursuer agent employs a deep deterministic policy gradient (DDPG) algorithm [30] to capture a moving target under imperfect information.

In Chapter 2, we showed significant performance advantages and robustness of this deep RL approach over a baseline pursuit strategy. Prior to the work presented in Chapter 2, no approach to pursuit or pursuit-evasion games addressed imperfect information, dynamic speed control, and speed overmatch (Table 1). Tackling challenges of uncertainty, speed control, and speed overmatch supports deployment of the resulting algorithm on real, unmanned systems in dynamic environments. In this case, the pursuer must learn anticipatory strategies to capture a speed-advantaged target and robust strategies to account for uncertainty. And results support that the pursuer does learn anticipatory strategies needed in order to capture a speed-advantaged target. In Chapter 2, based on [37], the authors hypothesized machine strategies through manual inspection and characterized performance in terms of user-proposed environmental conditions.

In this research, we make important observations across behaviors, game outcomes, environmental conditions, and human relatability. We demonstrate utility of machine introspection over action-only alternatives (Figure 17), uncover aggregate human-relatable strategies in terms of explainable initial conditions (Figure 20 & Figure 21), and discuss how these relationships can be exploited for online performance prediction. Action-only analysis refers to examination of only the actions that the machine took over time, without consideration of the activation patterns leading up to the action determination. Lastly, we define a novel measure of machine *commitment* based on the volatility of activation expressions, measured as a function of rolling-horizon Shannon entropy and analyze its correlation to resulting machine performance (Figure 24).

Section 1: Defining Machine Behavior

In general, we, and others [25], define machine behavior as the activation patterns inherent to outputs (y_i) in a neural network. A neural network is typically composed of a series of layers, wherein each node in subsequent layers receives an input signal from its previous layer ($\sum_{j=1}^{n} w_j x_j$), whereby their weights (w_j) are activated, via a non-linearity function, to produce corresponding output(s) to the next layer in the network until the final output layer is reached. This process is shown in Figure 14 for a single node. During online inference while the trained agent is competing in the pursuit game, input signals arrive, starting with the 18-dimensional pursuer and target state observation described in Chapter 2 Section 2.2, weights are applied (that were learned during training) using a linear operation to combine inputs to produce a single input, and an activation function (Figure 19) is applied to the single input to determine the output of the node. The output from this layer is fed to the subsequent layer until the final output layer (consisting of heading & acceleration actions for our 2-d pursuit application) are reached. The specific network structure and the outputs used in our behavior definition are described in Figure 15.

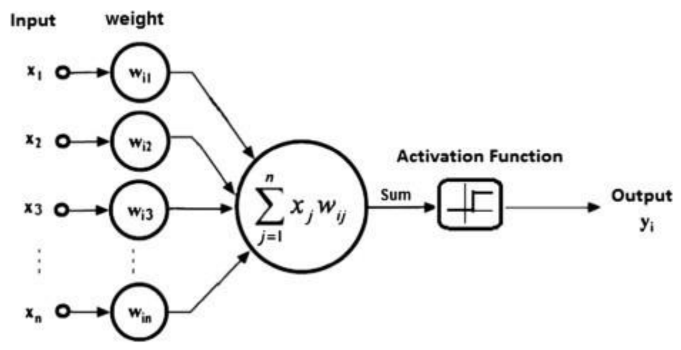


Figure 14 Visual depiction of neural network basic functions at a single node level. [38]

The neural network learns particular weight parameters in the network during a training phase, using the DDPG algorithm described in Chapter 2 Table 4 over 30,000 training episodes (2-d pursuit games) and associated target outputs (rewards). Weights are optimized so they maximize the expected future rewards, as defined in Chapter 2 Section 2.5 for our pursuit problem, encouraging target capture. Once the network is trained, we "freeze" weights associated with each node. That is, when we provide the network particular inputs, we get the same outputs (hence the "deterministic" qualifier for the DDPG method).

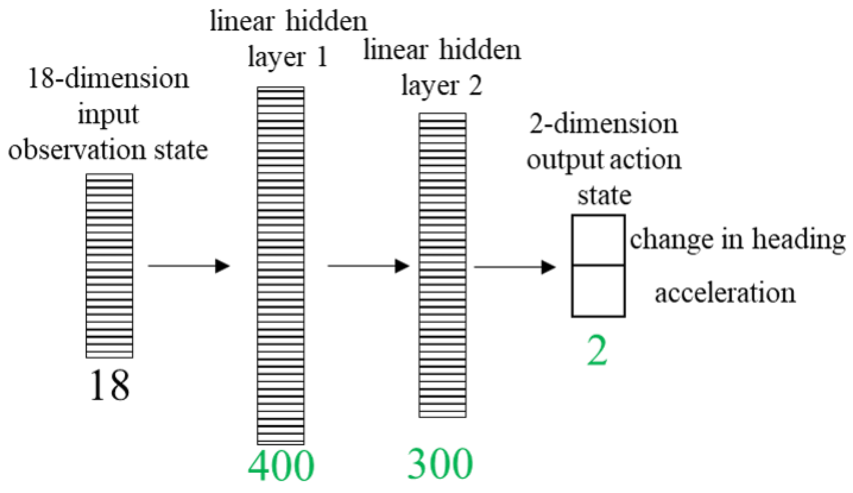


Figure 15 The network structure that determines our pursuit agent action consists of an 18-dimensional observation state input, two linear hidden layers, and a final output layer. A total of 702 outputs from the hidden layers, shown in green, are included in our behavior definition.

In our network, we have an input layer (the Observation, as described in Chapter 2 Section 2.2), two hidden layers, and an output layer (the two-dimensional action, as described in Chapter 2 Section 2.3). Here, when we refer to dimension, we refer to the number of nodes that are present in each layer. Each node works as described in Figure 14. We use batch normalization and the Rectified Linear Unit (ReLU) activation function (Figure 16 left) for our hidden layers and the hyberbolic tangent (tanh) activation function (Figure 16 right) on the final output layer, consistent with choices made in the successful DDPG implementation in [30]. The activation function is applied to the weighted sum, prior to determining the final node output, as shown in Figure 14.

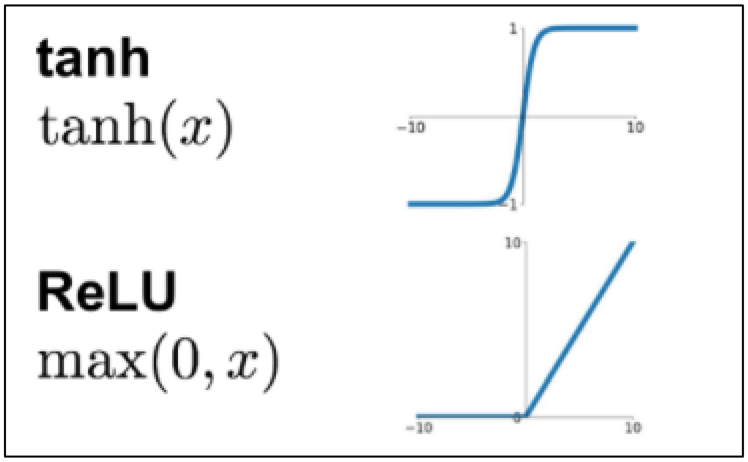


Figure 16 Activation functions used in our network

When conducting introspection (analysis of the internal machine activation patterns / behaviors), we consider our machine behavior definition as the 702 activations (400 (output from hidden layer 1) + 300 (output from hidden layer 2) + 2 (output from hidden layer 3) nodal outputs) in the network, specifically, the outputs of the layer weights through their respective ReLU and tanh activations. In other words, the y_i outputs from each i^{th} node (as shown in Figure 14) in the hidden layers make up our behavior (shown in green in Figure 15).

Similar behaviors (activation patterns) are then clustered together into strategies, many of which are interpretable by humans and in terms of conditions. Moreover, fluctuations in neuron channels are examined by our commitment measure, which is based on Shannon entropy.

Section 2: UMAP Algorithm

To abstract the strategies of a trained Deep RL agent, we employ a dimension-reduction technique on the activation patterns as the agent negotiates the pursuit task. More precisely, we define a single-machine behavior as the underlying neurons (nodes) activated during inference over the entire game duration. Then, we reduce the dimensionality of each game episode using the UMAP algorithm [29]. UMAP, developed in 2018, is a graph-based method for dimension reduction that is rooted in Riemannian geometry and algebraic topology; it is robust for use on sparse, time-series, and high-dimensional data. Here, UMAP takes as input all 702 activations for each time step and maps them into a two-dimensional space, amenable to human inspection. That is, we reduce 351,000 data points from each pursuit game (500 time steps x 702 activations) to just two values, encoding relevant information for machine-behavior analysis. The resulting illustration of these games, color-coded by the game outcome, is shown in Figure 17 (left).

Interestingly, when we apply the same dimension-reduction technique to just the two-dimensional action space (heading & acceleration outputs from the neural network) over the course of the game, we lose separation over the game outcomes

(Figure 17 right). In the case of Figure 17 (right), we analyze 1000 data points per game that contain the two-dimensional action and heading actions that occur at each of the 500 time steps. **In summary, we can learn more from looking at machine "thought processes" (activations) over time (351,000 data points per game) rather than just the actions themselves (1,000 data points per game), even though only the actions affect the environment.** This is a very important result, motivating further XAI research into analyzing machine behaviors for insights into machine competency.

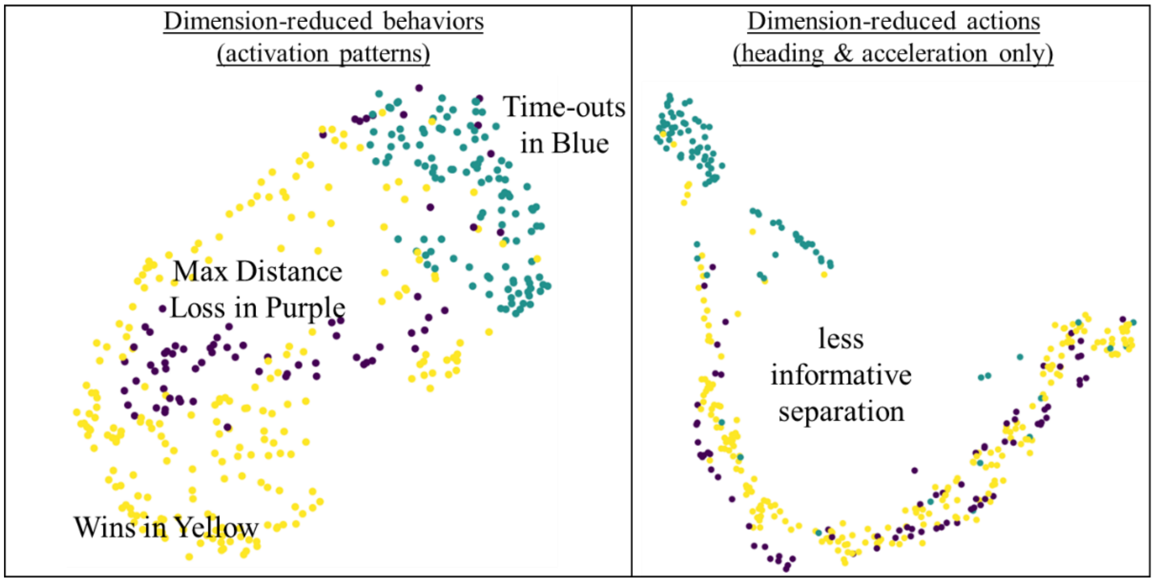


Figure 17 Visual inspection reveals that UMAP-embedded behaviors (left) encode more information relevant to the game outcome (color) than just the actions (right). Understanding how the agent thought about its actions revealed more relevant performance information than actions alone.

Complete details for the UMAP algorithm and theoretical foundations are available in [29]. UMAP is a graph-based dimension-reduction technique that aims to preserve local structure in data through manifold approximation. A *manifold* is a topological space that resembles Euclidean space around each point in the space [39]. In our 2-d pursuit game application, we used UMAP to reduce 351,000 activations associated with each game into a two-dimensional space. The behavior for each game is represented by 702 neurons changing over 500 time steps, yielding 351,000 data points. The resulting UMAP dimension reduction from 351,000 data points to two dimensions supported behavior visualization and interpretability with respect to other game parameters, such as outcome, conditions, and strategy abstractions.

The UMAP algorithm can be broken down into two phases: 1) relevant weighted-graph construction and 2) low-dimensional layout optimization. In Phase 1, we assume that data are uniformly distributed on some manifold, despite the fact that not all data are distributed uniformly. To create such a space, we employ Riemannian metrics [40], which take as input a pair of tangent vectors at a point (in our case, a node connecting two edges in an abstracted graph) and produce a scalar that characterizes the length and angle between the vectors, similar to a dot product. We surmise a manifold where data are separated by varying Riemannian metrics, forcing uniformity. That is, we craft a metric that is defined dynamically to force uniformity between data placed into the manifold. Since we have forced uniformity in this approximate manifold, we now have data separated with different metrics that we wish to merge into consistent global structure and do so using local fuzzy simplicial sets constructed using a k-nearest neighbor descent algorithm [41]. In our 2-d pursuit game application, we chose a k of 100, where 100 closest games are evaluated in the nearest neighbor algorithm. Then, we constructed a weighted directed graph over this manifold using a weight function described in Eq. 10 where we compute the set of k nearest neighbors of each data point x_i under the metric d where d represents a customized metric parameterized by ρ_i and σ_i that ensures that at least one other point

(x_i-) of the k nearest neighbors considered is connected by an arc in the graph to x_i. Here, x_i- refers to any point other than x_i. In other words, we choose the distance metric to be loose enough so that every point (node) is connected to at least one other point by choosing the appropriate ρ_i and σ_i that guarantee this property.

$$w\big((x_i, x_i\text{-})\big) = \exp\left(\frac{-\,max(0, d(x_i, x_i\text{-}) - \rho_i)}{\sigma_i}\right) \qquad \text{(Eq. 10)}$$

Ensuring connectedness has considerable computational benefits over other dimension-reduction techniques, such as t-distributed Stochastic Neighbor Embedding (t-SNE); for example, UMAP takes 87 seconds to perform dimension reduction on the Modified National Institute of Standards and Technology (MNIST) data set while t-SNE takes 1450 seconds [29]. The MNIST data set [42] is a large database of handwritten digits that is commonly used for training machine learning-based image processing systems to correctly classify the digit that was written. Finally, the UMAP algorithm optimizes the total cross entropy between the higher-dimensional graph and the graph projected into lower-dimensional space. Cross entropy [29] measures the difference between two probability distributions. This step guarantees matching of the dimensionality-reduced topology as closely as possible to the overall topology of the original data, thus providing a good low-dimensional representation.

Section 3: UMAP Dimension Reduction & Visualization of Deep RL Behaviors

In Table 10, we share the procedure for abstracting machine behaviors into strategy clusters. In Figure 17 (left), we see natural groupings into three color-coded regions corresponding to wins, max distance losses, and time-out losses.

Table 10 Procedural Pseudocode for Behavior Analysis

Parameters: Number of clusters (nk), as determined by "elbow" method illustrated in **Figure 18**

```
1. Load in activation patterns, game outcomes, and conditions
2. Perform UMAP dimensionality reduction
3. Display UMAP results color-coded by game outcome (Figure 17)
4. Display UMAP results color-coded by condition (Figure 21)
5. Cluster behaviors using kmeans over nk clusters to assign strategies
6. Display UMAP results color-coded by strategy cluster (Figure 20)
7. Display strategy-outcome plot (Figure 19)
8. Organize game visualizations into strategy-delineated file structure
9. Manually inspect strategy file folders for human-interpretable strategy labels (labels provided in Figure 20 and
sample games provided in Appendix)
```

Section 4: Abstracting behaviors into human-relatable strategies.

Now that we have a compact representation of machine behaviors in two-dimensional UMAP-reduced space (Figure 17 Left), we define strategies. *Strategies* are clusters of machine behaviors. We grouped behaviors into 13 clusters using a k-means algorithm and analyzed the game trajectories across each strategy. k-means clustering [43] organizes data into k groups, where each data point belongs to a single cluster with the nearest center. In our application, we clustered the pursuit games after they were reduced to two dimensions using UMAP. For k-means clustering, the number of clusters k is given as a parameter to the algorithm. Then the algorithm performs a heuristic routine to minimize within-cluster variances (Eq. 11) using the heuristic approach described next where the distance metric is also provided as an argument. In our case, we used the Euclidean distance.

$$\underset{\boldsymbol{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|\boldsymbol{x} - u_i\|^2 \ \ (Eq. 11)$$

In Eq. 11, $\boldsymbol{x}$ are the data points organized into k clusters noted by the sets S_i and u_i is the centroid of the i^{th} cluster. Heuristic approaches vary for solving this NP-hard problem. Here, we employed a `TimeSeriesKMeans` algorithm from the `tslearn.clustering` toolbox with default settings, Euclidean distance metric, and a given k [44].

First, the standard algorithm initializes assignments randomly by choosing k points at random as the cluster center. Next, it assigns each data point to the cluster with the nearest center. Finally, it re-computes the cluster centroids and repeats the assignment step until the assignments no longer change or a maximum iteration is reached.

We chose k based on two factors: the inertia and the resulting mixture of game outcomes. *Inertia* is the sum of distances of data points to their closest cluster center. In other words, it is the value found in Eq. 11 for particular assignments, $\sum_{i=1}^{k}\sum_{x\in S_i}\|\boldsymbol{x}-u_i\|^2$. To determine the range of acceptable numbers of clusters, we performed an "elbow analysis" [43] over the inertia attribute. The plot in Figure 18 shows how the inertia varied based on the different numbers of clusters given.

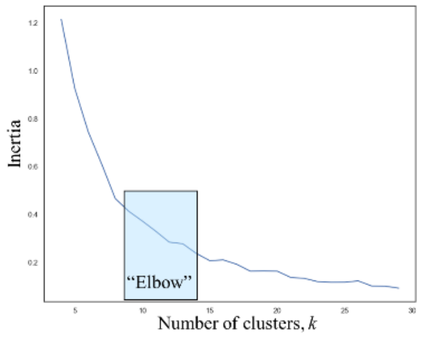


Figure 18 Using the "elbow method", we find a range of acceptable numbers of clusters, k, highlighted in blue

Similar to "knee of a curve" analysis, the goal of the elbow method is to identify a region of diminishing returns. In this case the elbow appears somewhere between $k=9$ and $k=14$ by visual inspection. To cut down our selection of k further, we examined the breakdown of game outcomes for each of the six k candidates and found that $k=13$ had the most homogeneous clusters with respect to game outcome (Figure 19). That is, we visually inspected the resulting mixtures of strategies and associated game outcomes and found that $k=13$ had the best separation over game outcomes. For example, strategies 3, 9, and 10 are solid green indicating those strategies led to 100% Win outcomes (homogeneous mixture). Conversely, Strategy 0 is a heterogeneous mixture over game outcomes, including wins (green), time-out losses (orange), and max-dist losses (blue). Selection of k is important for behavior analysis as it can also serve as a tuning parameter for the granularity of strategies, explored further by ongoing research summarized in [45].

As shown in Figure 20**, we see natural separation over strategies that are able to be assigned human-relatable labels**. Many of the strategies correspond to the manual strategies determined in Chapter 2. Moreover, we see separation of strategies over wins (Figure 19). That is, some strategies, like Strategy 9: "L-shaped maneuvers" and Strategy 3: "head-on approaches" are more likely to result in a winning outcome than others, such as "Strategy 12: Large distance traveled, no capture" (Figure 19, Figure 20). Thus, **if we know the strategy employed by the underlying AI system, we can characterize expected performance.** This result further motivates an online predictive capability for strategies, as developed in Chapter 4.

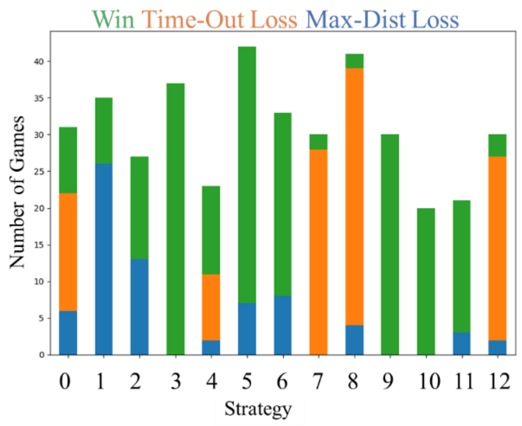


Figure 19 Strategies are composed of different, sometimes homogeneous, game outcome mixtures.

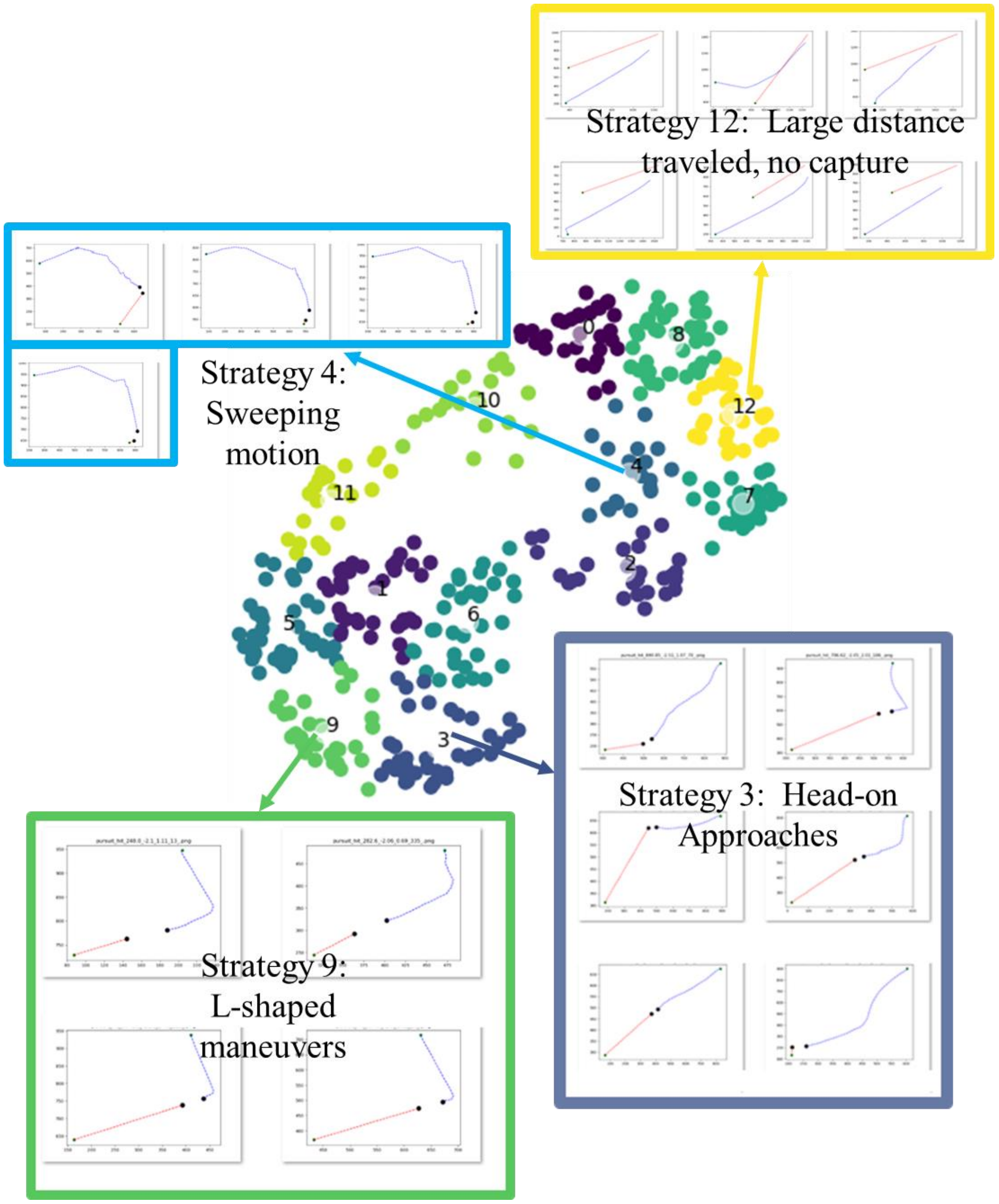


Figure 20 Clustering behaviors led to human-interpretable separation over strategy clusters. Additional sample games for each of the 12 strategies are available in Appendix B.

Section 5: Determining effects of environmental conditions on strategies and performance

Next, we examine the relationships between three expert-provided environmental conditions, strategies, and performance. The three expert-proposed conditions for this analysis are “initial angle to target”, initial distance to target”, and “maximum speed of the target”, as proposed and analyzed in Chapter 2 based on familiarity with the problem and resulting performance plotted with respect to each proposed condition. Instinctively, we expect these conditions to affect the game outcome and relate to machine behaviors; the plots in Figure 21 confirm our intuition.

For example, when we examine the collection of points associated with max distance losses (purple points in Figure 17), we observe that the pursuer’s initial position is behind the target (Figure 21 Left), far away from the target (Figure 21 Center), and at a speed disadvantage (Figure 21 Right). Here, we classify an initial angle as “behind the target” in the approximate range [-1, 1] radians (-60 to 60 deg). Moreover, when we examine cases where the agent failed due exceeding the maximum time limit (blue points in Figure 17), we see many instances of high target speed (Figure 21 Right). Lastly, just as Strategy 3 (“head-on approaches”) visually presented many cases where the target was initially positioned ahead of the pursuer (Figure 20), we see favorable angles in that region (Figure 21 Left) which correspond to 100% Win outcomes (yellow points in Figure 17). Here, we classify an initial angle as “favorable” in the range $[-\pi, -2] \cup [2, \pi]$ radians.

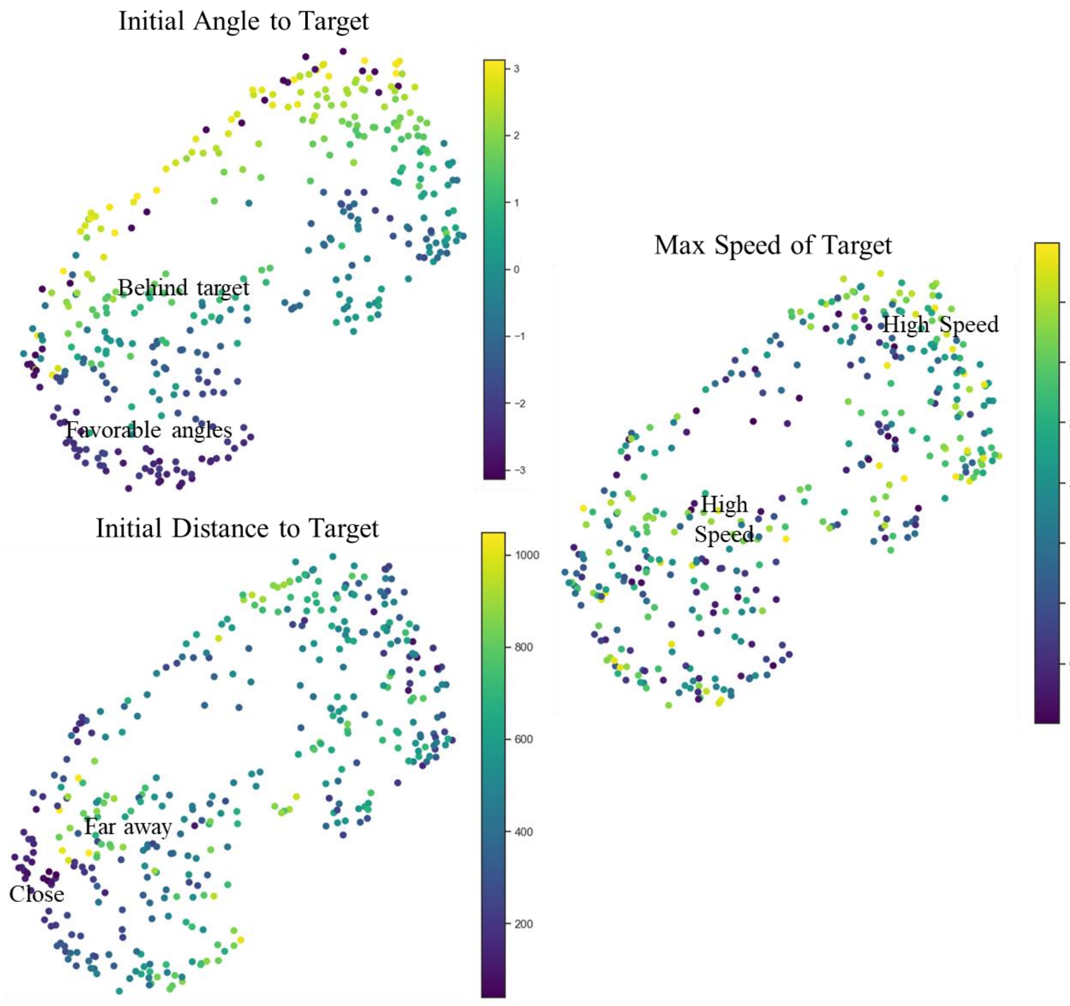


Figure 21 Behaviors relate to expert-proposed conditions intuitively.

Section 6: Defining machine commitment and determining commitment effects on game outcome

Now, we define and analyze a novel measure of machine commitment generalizable to all deep learning time-series applications. The goal of a commitment measure is to capture volatility in machine behavior, where volatility captures the fluctuation in activation patterns. We use Shannon entropy (Eq. 12) as a measure of volatility. We hypothesize that machine “indecision” leads to less favorable outcomes. Here, we define “indecision” as high volatility in activations over time. This indicates that the machine has (widely) volatile behavior, analogous to a human exhibiting wavering behavior, switching from one strategy to another without commitment to a specific strategy. However, we note, perhaps conversely, that *empowerment* (Section 7), which measures the act of leaving one's options open [46] [47] [48], has shown positive effects on machine performance as well as curiosity [49].

To frame a discussion around machine commitment, we showcase different activation patterns for a sample of ten games, chosen arbitrarily, from our 2-d pursuit application in Figure 22. We refer to each particular hidden node in the network as a *neuron* and each *neuron channel* represents the outputs (activations) from each of the 702 nodes over time. In Figure 22, we show these 702 neuron channels (subplot y-axis) over 500 time steps (sub-plot x-axis) for 10 sample 2-d pursuit games (subplots). We display how the raw machine activation patterns (before dimension reduction) change over time using color intensity and notice how they differ greatly between games. For example, just two neuron channels of 702 are active in Game 7 while seven are active in Game 4; in some cases, like Game 6, neurons "turn on" and stay on; whereas in others, like in Game 9, they turn on and off at various times in the game. Not surprisingly [50], we see very sparse utilization across the neuron channels. That is, few neuron channels are active for any one game, and this is a common observation for deep learning approaches [50]. This is analogous to using few neurons in your brain to think about something. We use a measure of volatility to understand the fluctuations inherent to these behaviors, and we leverage strategy clustering to make sense of these patterns at a higher level of abstraction.

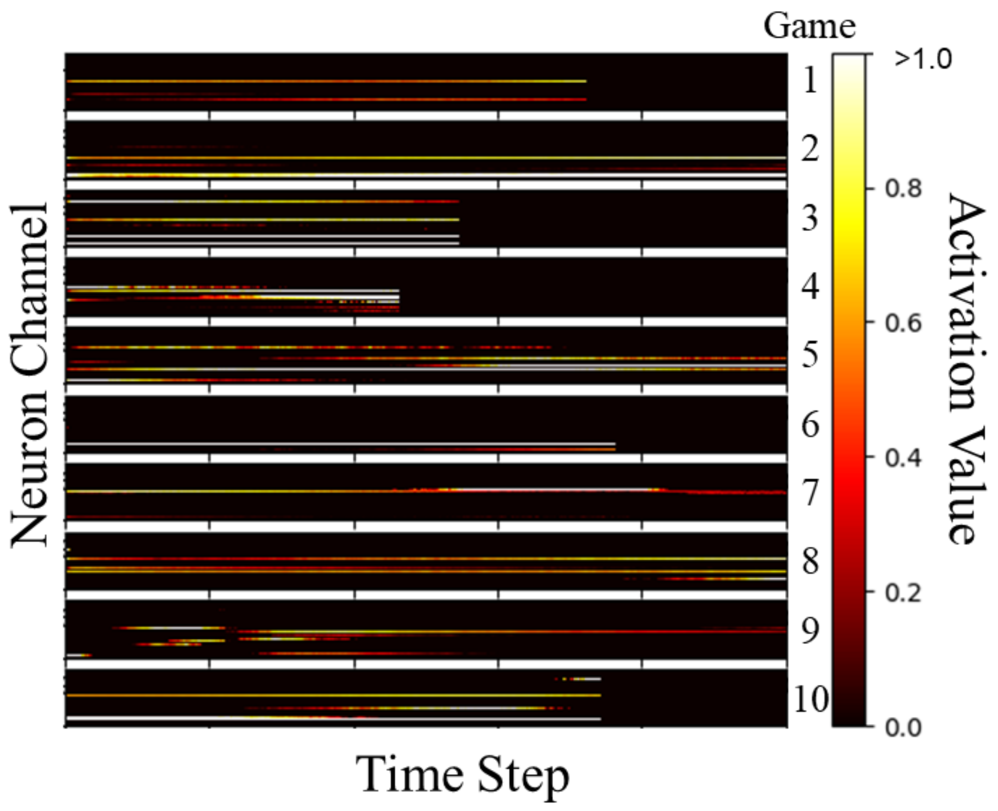


Figure 22 Neuron channels are sparsely utilized and vary throughout each game. Each subplot represents a separate game episode.

Shannon entropy (Eq. 12) has long been used as a measure of the amount of uncertainty inherent to a variable over time [51].

$$-\sum_{t\,\in\,\tau} P(x_t)\log P(x_t) \qquad\qquad (\text{Eq. }12)$$

In our application, we are studying the Shannon Entropy of each of the neuron channels as they evolve over the course of the 2-d pursuit game. For each neuron channel, each $P(x_t)$ is the activation value (output from the node) for a time t in a time horizon τ, such as {10, 20, 30, 40}. The entropy is calculated by the expression given in Eq. 12 for each node channel by using the `entropy` function available in the `scipy.stats` Python library [52].

We capture a rolling-horizon entropy score from the current horizon planning start time to the end of the game as the agent progresses through the game. As we are performing analysis after the games have been completed, we have perfect foresight to the end of the game at each time step for our behavior analysis. We note that rolling-horizon entropy knowledge into the future is not information that is available while the agent is online during the game execution itself and would have to be obtained similarly to simulating expected action interactions with the environment. That is, the machine does not have perfect foresight into its future behavior; we are analyzing its behavior throughout the game after the game has ended. An online predictive capability would need to be designed in order to anticipate future behavior and predict machine commitment *in situ* in order to gain insights into machine commitment while the machine was performing its task. We also note that for games with long or unknown durations, a shorter planning horizon would need to be considered for computational tractability. However, we study perfect foresight Shannon entropy here to determine its utility for a commitment attribute discussed in more detail in the next section.

We propose a novel definition of machine commitment, wherein we measure volatility of the neurons over receding (or rolling, based on the length of the game) time horizons. Shannon entropy describes the level of volatility and information gained from activation values as they change over the course of the game.

$$Commitment = \max_{\tau \in \boldsymbol{T}} \sum_{t \in \boldsymbol{\tau}} \sum_{n \in \boldsymbol{N}} P(x_{tn}) \log P(x_{tn}) \qquad (Eq. 13)$$

$\boldsymbol{T}$ consists of the set of all time horizons under evaluation. For example, for a long game with a rolling horizon entropy commitment calculation over a time horizon step size of 10 and horizon of 30, $= \{\{0, 10, 20, 30\}, \{10, 20, 30, 40\}, \{20, 30, 40, 50\}, \dots\}$, where τ is a set of times in $\boldsymbol{T}$, such as $\{10, 20, 30, 40\}$. In Eq. 13, to calculate the level of machine commitment in each game, we sum over the Shannon entropy [53] of the neurons n in the set of all 702 neurons N. For each neuron channel n, each $P(x_{tn})$ is the activation value (output from the node) for a time *t* in time horizon τ. We characterize commitment as the maximum negative entropy value over all of the horizons τ. For example, if the highest value was associated with $\tau^* = \{10, 20, 30, 40\}$, then it would correspond to the second element in $\boldsymbol{T}$, where *t* indexes each element of τ^*.

We note that this is a worst-case assessment of volatility because the entropy of two simultaneous events is no more than the sum of the entropies of each individual event [51]. In other words, the entropy of one neuron channel added with the entropy of another neuron channel is an upper bound on their joint entropy. Thus, we can sum entropies over all the neuron channels to obtain an upper bound on their joint entropy. While not studied in this research, we note further that the sum of entropies is *equivalent* to their joint entropy if all events (separated along neuron channels) are independent [51]. Next, we take the maximum value over all times to capture the worst-case volatility value for each game. And finally, since we are aiming to measure commitment (opposite of volatility), we take the negative of the volatility measure provided by the Shannon entropy.

In our 2-d pursuit game application, we chose a planning step size of 20 seconds for a game with 500 seconds (4% increments). That is, $\boldsymbol{T}$ was equal to the set of time horizons, $\{\{0, 20, 40, \ldots T_f\}, \{20, 40, 60 \ldots, T_f\}, \{40, 60, 80, \ldots, T_f\}, \ldots\}$, and the set of time horizons was truncated according to T_f. The rolling-horizon approach is extensive to long games with a finite planning horizon less than that of the entire game, so that it can scale computationally. However, for this analysis, we assume perfect foresight knowledge of the machine behaviors until the end of the game and examine the resulting receding-horizon volatility over time. That is, the planning period gets shorter as we step through the game.

Figure 23 shows the volatility portion (Eq. 14) of the commitment measure, which is computed for each time horizon $\tau \epsilon \boldsymbol{T}$. Each line plot represents a separate game episode and is color-coded by game outcome; each point in the line represents the volatility measure for a certain time horizon τ. The receding-horizon volatility (y-axis) captures the fluctuation in activation patterns from the current planning period start time (x-axis) to the end of the game. Since the planning period is receding (getting shorter in duration) and we are adding entropy (positive value) at every time step in the receding horizon, the volatility monotonically decreases over time. This is because fewer and fewer time steps are included in the calculation as we get closer to the end of the game.

$$\sum_{n \in N} \sum_{t \in \boldsymbol{\tau}} -P(x_{\tau n}) \log P(x_{\tau n}) \quad (Eq. 14)$$

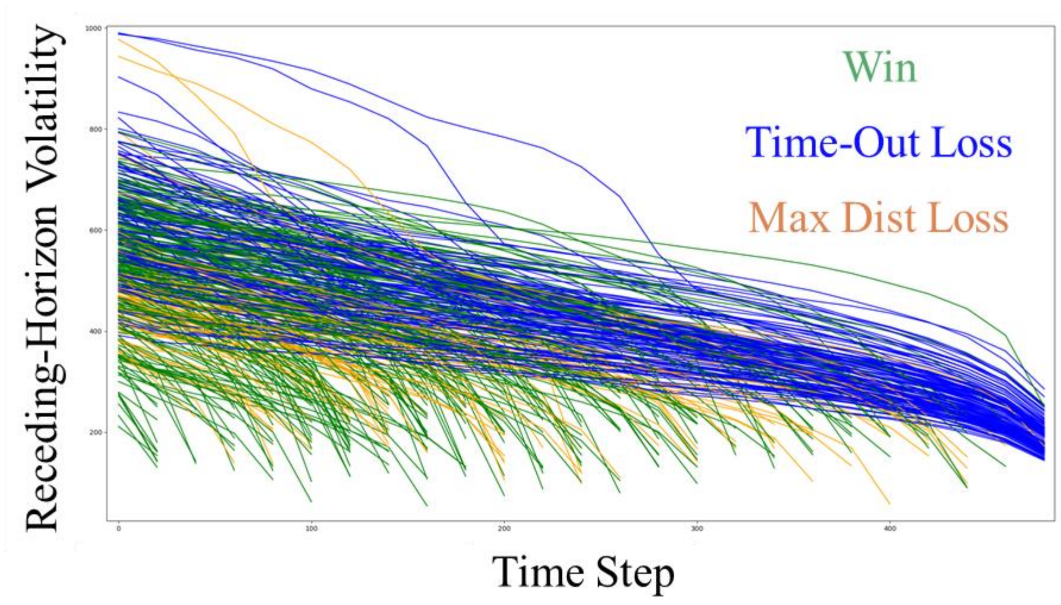


Figure 23 Receding-horizon volatility measured to the end of the game monotonically decreases as the agent progresses through the game, as expected.

In Figure 23, we note that games with "win" outcomes (green) tend to have lower receding-horizon volatility at the start and over the course of the game than the other game outcomes, decreasing significantly toward the ends of the games (lowest green lines on the plot). We also note that games with "time-out loss" outcomes tend to be flatter with respect to their receding-horizon volatility than the other game outcomes.

As an alternative to entropy, we also investigated the standard deviation of machine behavior over time as a volatility measure suitable for basing a commitment definition. However, as standard deviation accounts for the fluctuation in the actual value of a variable, it was not as well-suited for the desired measure of volatility. Standard deviation measures the spread of the data in terms of actual values, which we do not care about. For example, for a bimodal distribution of activation values, as the distance between peaks widens, the standard deviation increases. Conversely, the

Shannon entropy stays the same. In our application, this is analogous to a cyclical pattern of activation values over time, which we want to characterize as more committed than a random expression of activation values over time. Shannon entropy characterizes this as desired while standard deviation would not. In addition to this distinction, we also examined the resulting commitment values and found that they did not vary significantly between game outcomes, rendering it a worse alternative to characterize machine commitment.

Finally, we examine the relationship between commitment and game outcome by taking the negative of the maximum receding-horizon volatility (shown as box plots in Figure 24 with respect to game outcomes and in Figure 25 with respect to strategies). In the case of perfect foresight and full-game receding planning horizons, this is the first (maximum) value in Figure 23. In Figure 24, we observe that when the commitment is above a value of -342 (unitless), 100% of the games result in a Win.

In order to test the significance of our commitment results, we employed a Mann-Whitney U test, which tests whether one population of observations is greater than another [54]. In particular we used a Mann-Whitney U test to determine whether the commitment distribution for any game outcome was significantly greater than another. The results from the test revealed the distribution of commitment levels for the Win outcome was significantly greater than those of the Time-Out Loss (*p-value = 1e-3*) and Max Dist Loss (*p-value = 2e-8*) game outcomes.

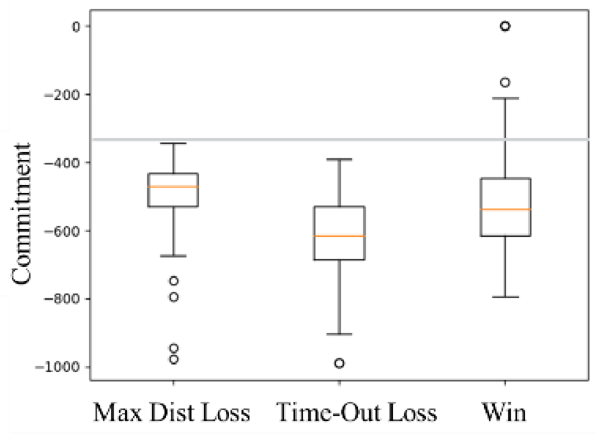


Figure 24 Games with Win outcomes have significantly higher commitment than games with Loss outcomes. Games with commitment values above -342 always resulted in a Win.

Here, again (like strategies), we determine that commitment is a behavior-based predictor of expected performance that can be exploited during online machine control; **we also postulate that when commitment is estimated as high, humans can gain more trust in machine behaviors, because machines will remain committed to a certain strategy.** For example, if the "L-shaped" strategy was estimated with high confidence and commitment was also accurately estimated to be high, then the human teammate could anticipate a 90-degree turn from its machine partner prior to its occurrence; this discussion and its derivatives have many redeeming benefits for human-machine teaming.

In Figure 25, we show the commitment distributions partitioned by each strategy group. Each game is assigned a strategy using the procedure described in Table 10 and the results for each strategy assignment are provided visually in Figure 20. We see interesting relationships between high commitment and Strategy 1 (as clustered by the method in Table 10), which surprisingly results in frequent maximum distance losses (described in detail in Chapter 2 Section 2.5), but executes straight line trajectories under impossible scenarios. We also note that Strategy 10 exhibits low commitment values but results in 100% Win outcomes, where many of the trajectories are jagged. Strategies 3 ("head-on approaches") and 9 ("L-shaped maneuvers"), however, also result in 100% Win outcomes and have higher commitment expressions than Strategy 10.

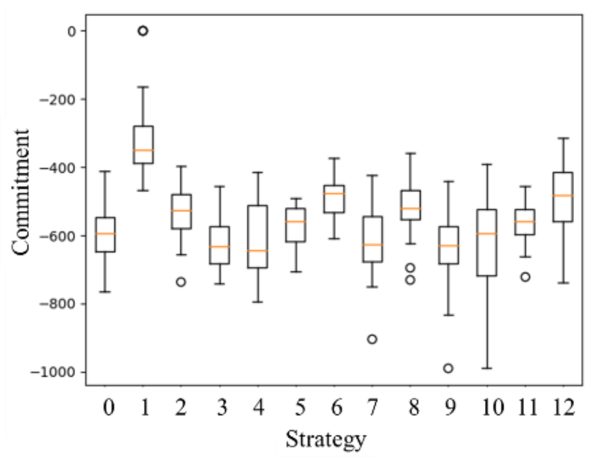


Figure 25 Strategies correspond to varying distributions of commitment values

In summary, the procedure for performing the commitment analysis is given in Table 11.

Table 11 Procedural Pseudocode for Commitment Analysis

```
Parameters:  Rolling horizon step size
1. Load in activation patterns and game outcomes
2. for each game, do
    3. initialize game rolling horizon entropy structure
    4. for each rolling horizon start time, do
        5. initialize entropy sum to 0
        6. for each neuron channel, do
            7. calculate the rolling horizon entropy for each time horizon
                $\sum_{n \in N} \sum_{t \in \tau} -P(x_{tn}) \log P(x_{tn}) \;\; for\ each\ \tau \in T$
            end for
        8. Store rolling horizon entropy in game rolling horizon entropy structure
        end for
    end for
9. Visualize rolling-horizon time steps (x-axis) and rolling horizon entropies for each time horizon (y-axis) (Figure 23)
10. Calculate commitment: $\max_{\tau \in T} \sum_{t \in \tau} \sum_{n \in N} P(x_{tn}) \log P(x_{tn})$
11. Visualize box plot of commitments by game outcome (Figure 24) and strategy (Figure 25)
12. Perform Mann Whitney U test on commitments for significance across game outcomes
```

Section 7: Empowerment

In order to mimic basic human curiosities and survival instincts, scientists have attempted to equip machines with intrinsic motivations. Approaches vary, but one intrinsic reward, *Empowerment*, seeks to maximize mutual information by taking actions which allow the machine to reach the largest number of future states within some planning horizon [48, 46, 47, 49]. *Mutual information* ($I(X;Y)$) (Eq. 15) measures dependence between random variables X and Y through their conditional probability distribution ($p_{(X,Y)}(x,y)$) and their corresponding marginal probability distributions ($p_X(x)$ and $p_y(y)$).

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p_{(X,Y)}(x,y)\log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_y(y)}\right) \quad (\text{Eq. } 15)$$

Empowerment seeks to maximize mutual information gathered across future actions over some planning horizon, annotated in Eq. 16.

$$E = \max_{p(a)} I(A;S) \quad (\text{Eq. } 16)$$

Here, A describes the actions taken over some planning horizon, $p(a)$ represents the probability distribution of actions, and S describes their respective observations. Empowerment seeks to find the probability distribution of actions that leads to the most diverse set of expected observations. In other words, when observations stay the same over time, as determined by mutual information scoring (Eq. 15), the machine is **not empowered**, as the mutual information would be zero. Conversely, if the machine observes widely varying observations, consistent with "exploration", then the machine **is empowered** as the mutual information is high [55].

Empowerment, however, does not explicitly consider the machine's behavior. Instead, it encourages different behaviors by motivating sensory exploration through its intrinsic reward. Commitment measures volatility of the underlying behaviors themselves. We note that commitment, defined as a function of behavior entropy, could serve as an alternate intrinsic reward when a desired attribute of the resulting agent is to be decisive. This does not directly conflict with the Empowerment intrinsic reward, as high commitment to a certain strategy may result in varying observations (high Empowerment). Future research could study the benefits of combining expectations of Empowerment and commitment as intrinsic rewards. However, intuitively, the Empowerment motto of "keeping one's options open" seems in opposition to commitment to a certain strategy.

Section 8: Interpretations.

In summary, we have shown considerable value in analysis of "thought processes" that govern machine behavior. For our 2-d pursuit application, we found that compact representations of activation patterns (*behaviors*) preserved more information relevant to machine performance than the machine's actions themselves. Moreover, we found that clusters of behaviors (*strategies*) separated naturally over pursuit trajectories that were relatable to humans and that strategies could be exploited as predictors for machine performance. Next, we found that machine behaviors related to expert-proposed environmental conditions intuitively. And finally, we defined machine *commitment*, and found that it was significantly higher in games with Win outcomes than Loss outcomes. As such, future research on predicting machine strategy and commitment *in situ* can offer greater insight and predictability into machine expected behavior and performance, thereby increasing user awareness and trust in “black box” ML systems.

Currently, the authors are applying these XAI introspective techniques to a supervised machine learning system for an image-based obstacle avoidance task, funded by the DARPA Competency-Aware Machine Learning (CAML) program [45], and the focus of the next chapter. Preliminary results attest to the generalizability and applicability of this approach to other AI systems and applications. In conclusion, research in this chapter contributes to increased AI interpretability and competency-awareness needed to increase trust and transparency of AI systems.

Chapter 4: Achieving Competency-Aware Machine Learning through Machine-Derived Conditions and Online Strategy & Performance Prediction

Section 0: Overview

Understanding the strengths and limitations of machine *competency* is important both to machines and to humans. Machine *competency* refers not only to the expected performance of a machine learning system, but also to the strategy (how) employed by the machine to perform its task. If humans understand machine performance, they gain trust into machines to perform tasks in high-probability success environments. Moreover, if humans understand machine strategies, they gain transparency into the machines and also increase trust in these otherwise "black box" systems. When humans understand how the machine is performing its task along with the expected performance, humans can adequately assess the competency of the underlying system. If the machine expected performance was high, but the strategy was random, a human would have less confidence in the overall competency of the system. For example, the expected capture performance (i.e. a percentage of 90%) for our Chapter 2 deep reinforcement learning pursuit game along with the expected strategy, such as "L-shaped maneuver", make up the machine's competency. Lastly, understanding the environmental conditions that affect machine competency helps provide additional context and insight into *why* the machine behaves the way it does, promoting additional trust in machines.

Competency assessment of machines is particularly important for the future of human-machine teaming applications. Example use cases include: manned aircraft working with unmanned aircraft; manned ground vehicles working with unmanned systems; human intelligence analysts working with machines to exploit intelligence imagery; human commanders working with machines to understand the battlespace and make informed decisions; unmanned transportation resources rendezvousing with manned surface vessels; and a multitude of other commercial and military human-machine teaming application areas.

Currently, most machine learning systems are trained on a static set of training data until a certain level of performance is achieved and the system is deemed effective by a subject matter expert (SME). Next, trained systems perform their assigned task, regardless of their competency to do so or to do well. Typical maintenance of AI/ML systems includes human monitoring of self-reported confidence scores, which are artifacts of machine experiences limited to the training data set and oftentimes not calibrated to probabilistic interpretation. Once a system is determined defective by a human performance monitor, a labor-intensive retraining phase ensues, and the labor-intensive deployment cycle repeats with fragile trust from the human teammate. Alternative state-of-the-art maintenance of AI/ML systems relies on online learning mechanisms; however, lifelong learning mechanisms are prone to catastrophic forgetting and lack human-verified performance validation, as the system is constantly evolving. In the future, we must ensure that trained systems maintain acceptable levels of performance even after long periods of deployment; and we need to recognize when such systems may be losing efficacy or under-performing on assigned tasks. Our MindfuL™ competency-assessment framework, described in this Chapter, provides a persistent and extensible capability for assessing competency of ML systems.

To exploit the relationships between environmental conditions and machine competency (strategies and performance) motivated by results in Chapters 2 & 3, we develop an online predictor system to share near real-time competency estimates with a user. Moreover, we aim to automatically derive conditions that affect machine competency so that our approach can be more broadly applicable to other applications and so that competency predictions can be traced back to identifiable conditions. Traceability supports explainability of our competency predictions in terms of environmental conditions for a human user. While Chapters 2 & 3 considered a time-series deep RL agent that took target observations as inputs, Chapter 4 considers a supervised ML system that performs a task on a per-instance basis with visual image data inputs. The obstacle recognition machine learning system takes images as input and determines whether an obstacle is present in the image. In this simple obstacle recognition application, the user is monitoring the machine results and associated competency. This is analogous to a human intelligence analyst determining whether to trust or not trust that a machine has processed an image for identifying objects of interest. And by applying introspective techniques in Chapter 3 to a different application and input data type, we show how our competency-awareness approaches generalize.

In this chapter, we develop an offline procedure for learning relationships between automatically-derived environmental conditions and machine learning competency that is generalizable to a large number of ML systems and applications. Specifically, we can work with any set of inputs and any deep learning system. There are no known limitations on applications to which this approach could generalize. In fact, this approach could assess performance even for systems that are not deep learning systems; however, the ability to predict the machine strategy would not be available, as the strategy approach works on the activation patterns of the underlying deep learning system.

Moreover, we employ and demonstrate an online competency-awareness system that predicts machine strategies and performance in near real-time and attributes competency predictions to environmental conditions. The resulting competency-aware machine learning (CAML) system is attached to a ML system under evaluation to act as a supervisor, passively determining whether the underlying system will maintain consistent behavior and performance under dynamically changing environments. The competency-aware system, dubbed MindfuL™ software, interacts with the user to safeguard underlying ML systems from being used inappropriately, and prioritizes tasks that require user intervention. For example, if a human and machine are tasked with the imagery exploitation (processing of an image for identifying objects of interest) of 1000 images to search for airplane objects, and the machine has never been trained on an image that is similar to 100 of those images, the competency-aware MindfuL™ system will flag these 100 images for manual exploitation. Conversely, the machine has been trained adequately, according to MindfuL™ environment similarity calculations, to exploit 900 of those images. However, 10 of them have low expected machine performance; the MindfuL™ system again flags these 10 images for manual inspection. Finally, the machine processes 890 of those images with high expected performance and associated condition and strategy predictions. The human can "spot check" any of these 890 instances and gain insights into why the machine determined there was or was not an airplane present in the image, such as “airport”, along with the associated strategy, such as “looked for wings”, thereby increasing transparency & trust in machine’s ability to perform the task without human verification.

Without MindfuL™ software, state-of-the-art machine automated target recognition systems process all 1000 images, regardless of whether they have been adequately trained. The confidence scores associated with machine predictions are prone to miscalibration, as they are based solely on the training data. And humans have no insights into why (conditions) or how (strategies) these “black boxes” arrived at their result. Competency-aware machine learning systems provide a built-in machine supervisor to quantify and qualify expected competency for each assigned task.

Chapter 4 focuses on the competency components (conditions, strategies, and performance) of the MindfuL™ system. The entire system is composed of 14 components, all of which have been developed and integrated for an obstacle recognition supervised ML application. The inputs and outputs of all components are summarized in Figure 26. However, detailed expositions for the MindfuL™ Memory Bank, Competency Statement Generator, Element Interpreter, Element Ingester, Competency User Interface (UI) and Information Analyzer components are not included in this dissertation. These supporting components are not considered novel and enable database storage and interfaces to the user.

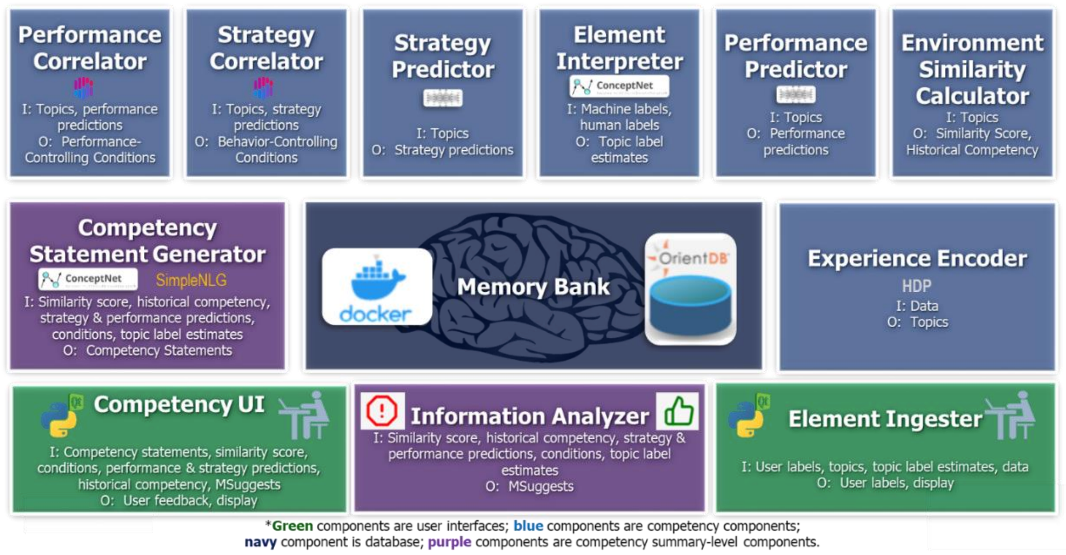

Performance Correlator
I: Topics, performance predictions
O: Performance-Controlling Conditions
Strategy Correlator
I: Topics, strategy predictions
O: Behavior-Controlling Conditions
Strategy Predictor
I: Topics
O: Strategy predictions
Element Interpreter
ConceptNet
I: Machine labels, human labels
O: Topic label estimates
Performance Predictor
I: Topics
O: Performance predictions
Environment Similarity Calculator
I: Topics
O: Similarity Score, Historical Competency
Competency Statement Generator
ConceptNet
SimpleNLG
I: Similarity score, historical competency, strategy & performance predictions, conditions, topic label estimates
O: Competency Statements
docker
Memory Bank
OrientDB
Experience Encoder
HDP
I: Data
O: Topics
Competency UI
I: Competency statements, similarity score, conditions, performance & strategy predictions, historical competency, MSuggests
O: User feedback, display
Information Analyzer
I: Similarity score, historical competency, strategy & performance predictions, conditions, topic label estimates
O: MSuggests
Element Ingester
I: User labels, topics, topic label estimates, data
O: User labels, display
*Green components are user interfaces; blue components are competency components;
navy component is database; purple components are competency summary-level components.

Figure 26 MindfuL™ online system input/output architecture

Chapter 4 is organized as follows. Section 1 describes our machine learning application and "ML system under evaluation"; this is the system whose competency is being assessed. Section 2 provides a description of our Experience Encoder, including how we automatically learn conditions from the environment using a Bayesian nonparametric (BNP) data processing method and store them compactly in a database (Memory Bank) for later reference. The application of BNP for enabling competency predictions is novel, expanding on prior research in crafting compact data representations [56] [57]. Section 3 describes how we leverage compact representations of the environment, stored in our Memory Bank, to determine the similarity of the current environment to past machine experiences. We also share how we leverage recognition of similar past environments to yield historical competency scores. This gives the user a sense of how many training experiences the machine has endured similar to the current one, and a distribution on both historical performance and historical strategies (historical competency) associated with those past experiences.

In Section 4, we discuss our approach to offline strategy and performance definitions, where we use the same UMAP dimension reduction and k-means clustering techniques from Chapter 3 on a new machine task and different deep learning approach. Section 5 discusses our approach to online strategy (how) prediction, where we take the outputs from the Experience Encoder and layer on a deep learning technique to learn strategy predictions. Similarly, in Section 6, we take in the same compact representation of the environment output from the Experience Encoder and learn performance (result) predictions. Next, in Section 7, we discuss how we layer on deep explainer methods (Performance Correlator & Strategy Correlator components) onto our predictors discussed in Sections 5 & 6 to determine which environmental conditions (why) were most influential on the predictions using SHapley Additive exPlanations (SHAP). In Section 8 we walk through the offline and online system distributed functional architectures, explaining how the MindfuL™ system is trained (offline) and how it works to process new data (online). And finally, in Section 9, we share interpretations of the aforementioned competency capabilities and implications to broader XAI research initiatives.

Throughout this chapter, reference will be made to our prototype Competency User Interface (UI), shown in Figure 27. In addition to being a debugging tool for software developers, the UI showcases our proof-of-concept competency assessments to a user. In the future, we will garner feedback from imagery analysts to see what competency information they find most useful for the tasks they share with their machine partners. Additionally, MindfuL™ software could be integrated into analyst workspaces, including widely used tools such as BAE Systems SOCET GXP® or

Government-owned iSpy software. Through the user interface, the user can provide additional semantic-label information to be used by the MindfuL™ system and can see machine competency information for the current image being processed.

Here, we briefly describe the competency information shared through the Competency UI. Outputs from the Environment Similarity Calculator component feed into the light blue boxes. In our imagery exploitation use case, this is the output that would correspond to the number of similar training experiences that the machine has seen previously; this output would help to flag the 100 samples that the machine should not process, as it has no similar experience with those 100 tasks. This measure gives the user an idea of how many times the machine has performed the task at-hand during its training.

Outputs from the Performance Predictor describe the expected machine performance to correctly exploit a particular image. Outputs from the Performance & Strategy SHAP components correspond to the conditions (why) that influence our competency predictions, such as “airport”. Output from the Strategy Predictor corresponds to the strategy (how) employed by the machine to perform the task, such as “looked for wings”. Through this interface, the user can observe the overall competency of the machine learning system and see estimates on the conditions that incited particular competence for the current task environment (image being processed). For example, overall competency could be low for cases where the entire image is spanned by “clouds” as a condition that negatively affects machine performance, especially if the ML system has not had training samples that included clouds.

Data acquired from a robot camera in simulation populate the image shown, which corresponds to the rest of the competency information displayed. Labels are assigned to different conditions output from the Experience Encoder either automatically or through the Element Ingester & Interpreter components, which are omitted from detailed discussion in this dissertation. Competency Statements are constructed by using open-source natural language generation software, which is also omitted from the scope of this dissertation. Finally, truth information is displayed both for performance, whether the ML was actually "blocked" or "free" according to LiDAR truth information, and for the strategy, as determined by UMAP transformation and k-means inference.

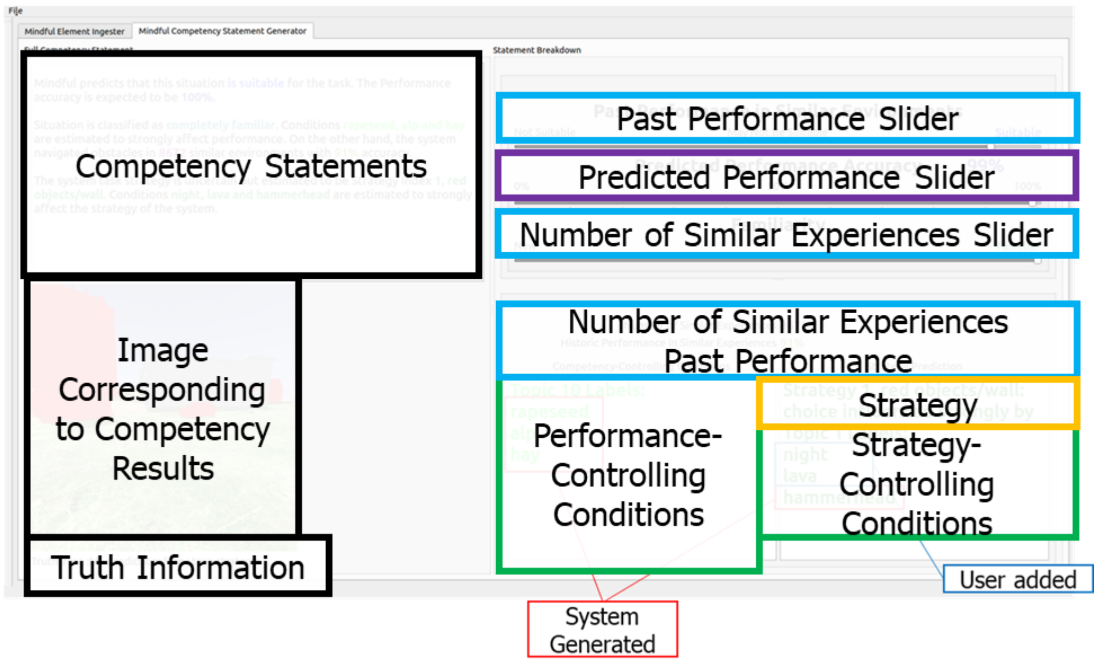
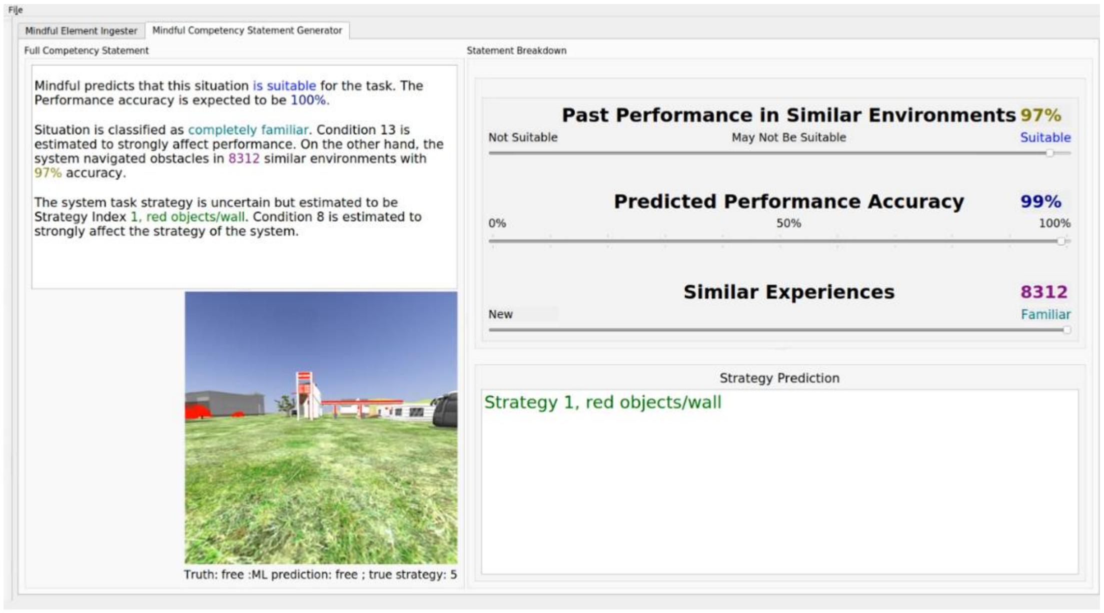

File
Mindful Element Ingester
Mindful Competency Statement Generator
Statement Breakdown
Competency Statements
Image Corresponding to Competency Results
Truth Information
Past Performance Slider
Predicted Performance Slider
Number of Similar Experiences Slider
Number of Similar Experiences Past Performance
Strategy
Performance-Controlling Conditions
Strategy-Controlling Conditions
User added
System Generated

Figure 27 Prototype Competency User Interface (UI)

Figure 28 shows a simplified version of our prototype competency user interface, easier to view without overlaid labels summarizing the information displayed. The competency information shown is an actual result from a MindfuL™

assessment of the associated image data shown. In this case, past performance and predicted performance are high, and the ML system has been trained on 8,312 images similar to this one; as such, MindfuL™ software summarizes to the user that the situation is "suitable" and "familiar". This would correspond to one of the 890 instances from our airplane exploitation use case, where the human can "feel good" about trusting the ML result. In the application displayed here, the machine has high expected performance of correctly determining that no obstacle is present in the image.



Mindful predicts that this situation is suitable for the task. The Performance accuracy is expected to be 100%.

Situation is classified as completely familiar. Condition 13 is estimated to strongly affect performance. On the other hand, the system navigated obstacles in 8312 similar environments with 97% accuracy.

The system task strategy is uncertain but estimated to be Strategy Index 1, red objects/wall. Condition 8 is estimated to strongly affect the strategy of the system.

Truth: free :ML prediction: free ; true strategy: 5

Past Performance in Similar Environments 97% Suitable

Predicted Performance Accuracy 99%

Similar Experiences 8312 Familiar

Strategy 1, red objects/wall

Figure 28 Simplified Prototype User Interface

<u>Section 1: Deep Supervised Machine Learning System under Evaluation</u>

Separate from the deep RL introduced in Chapter 2 and analyzed in Chapter 3, here we consider an agent that performs an image processing task on a per-image basis. That is, we consider a machine that processes visual data and determines a binary classification of whether there is an obstacle in the image or not. This task more generally supports ground robotic autonomous operation. Moreover, this task is similar to performing imagery exploitation to process images to determine whether an airplane is present or not, as described in Section 0. Obstacle recognition classification is a basic machine learning task that we are using as a proof-of-concept application. The task of determining whether an obstacle is present or not is not novel and is used as a simple task to use to show that our competency awareness approaches function properly.

The ML model is the same structure as the open-source Alexnet [58] model and was modified by replacing the final object classification layer with a binary classifier, where the output classification indicates a "blocked" or "free" estimate, based on whether an obstacle is present ("blocked") or not present ("free") in the image being processed. An illustration of the neural network structure is provided in Figure 29.

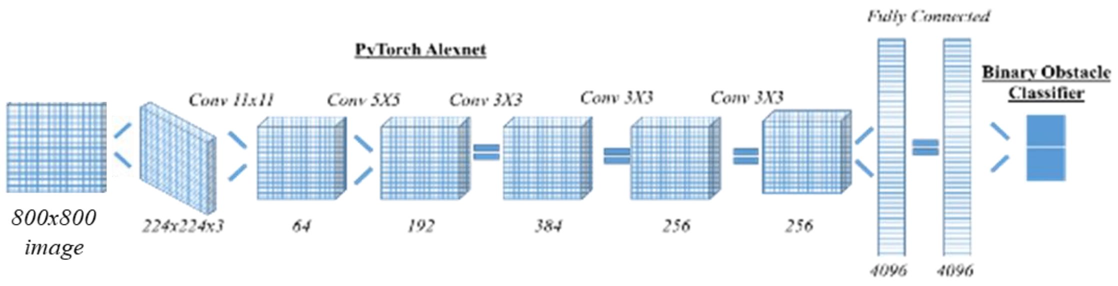


Figure 29 Deep Supervised Learning Model Structure

This Obstacle Classifier ML system under competency evaluation takes in an 800x800 pixel image generated from a Robot Operating System (ROS) and Gazebo simulation (described briefly in the next section) and outputs a classification of probability “blocked” or “free”, depending on whether the machine perceives an obstacle is present in the image. Sample visual data inputs for both “free” and “blocked” cases are provided in Figure 30. In a larger-scoped ground autonomy system, this obstacle recognition estimate could feed into an obstacle avoidance module.

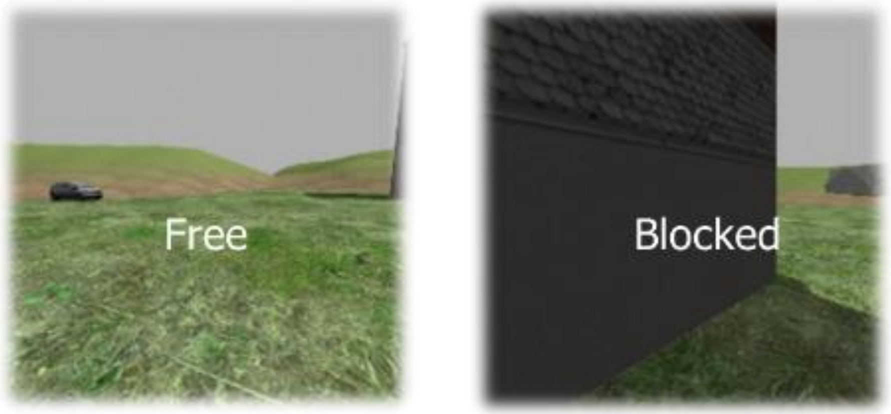


Figure 30 Example visual data inputs

A screenshot of our ROS/Gazebo simulation is shown in Figure 31. In the top frame, you see the ground robot traveling through a simulation environment. The camera is mounted on top of the robot along with a light detection and ranging (LiDAR) sensor. LiDAR inputs are used to determine true distances from the vehicle to the obstacle. The simulated LiDAR field of view is shown in blue. In the middle of the screenshot, the Laser truth associated with the LiDAR returns and the ML prediction from the Obstacle Classifier described above are shown.

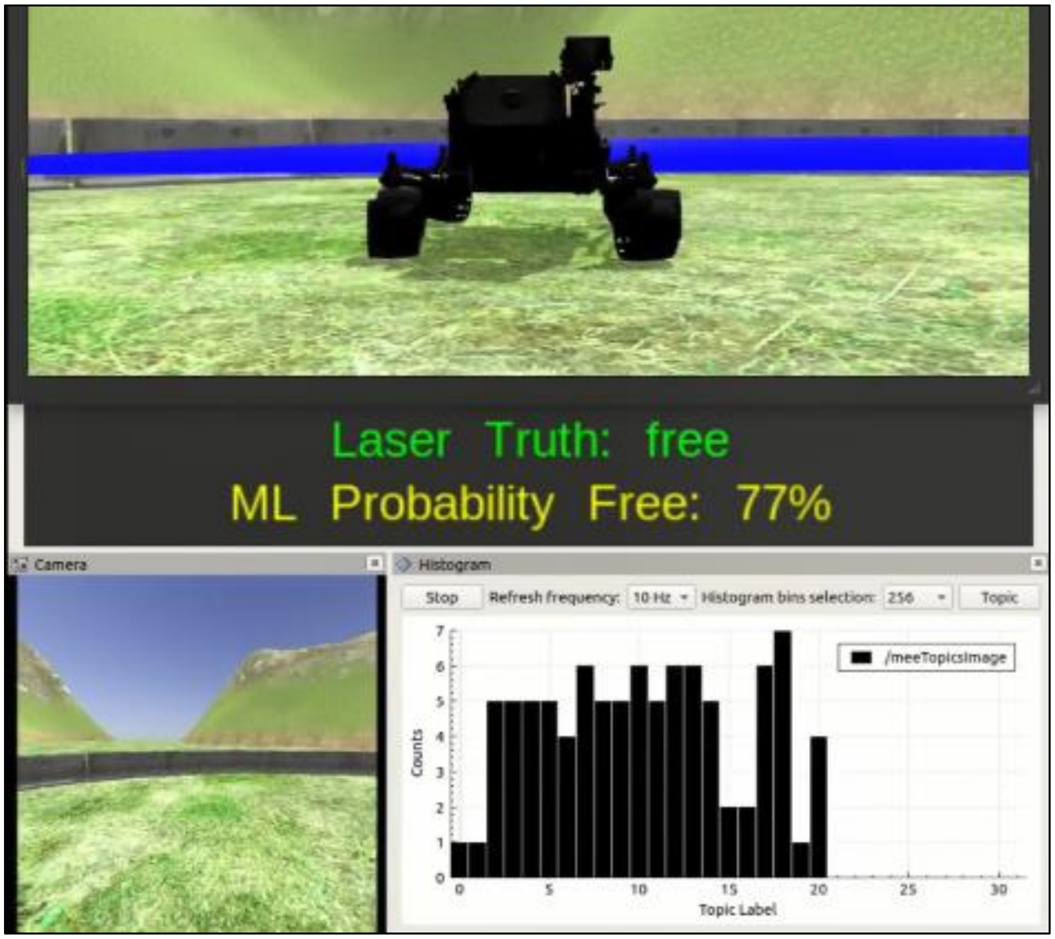


Figure 31 ROS/Gazebo Simulation Screenshot

The LiDAR laser truth is provided by the simulation and determines the distance to the nearest obstacle. It displays “free” if there is no obstacle within 6 meters; otherwise it displays “blocked”. The “ML Probability Free” is the output from the Obstacle Classifier neural network described above. In the bottom left of the screenshot we show the incoming visual data associated with the ML prediction. And in the bottom right of the screenshot, we show a topic distribution associated with the input image, where topic distributions are summaries over features extracted from the image. The next section discusses in detail how we determine compact topic distributions to describe the input image data.

Section 2: Automatic Derivation of Conditions via Bayesian Nonparametric Methods

In order to explain ML competency to the users, we apply a Bayesian nonparametric (BNP) method that summarizes the competency-controlling conditions while online. Data are taken as input, features are extracted, and features are organized into different topics (clusters) using a BNP approach, described in detail throughout this section. These topics (clusters) summarize the input data compactly into a histogram over combinations of features (topic definitions). In other words, we transform an image into a histogram over topics, where topics are defined as combinations of features extracted from the image. Ideally, our topics would separate over environmental features that are pertinent to the machine task and to machine competency. For our obstacle recognition task, this could include things like specific obstacles, such as a single topic for "building", "wall", or "clear grassy foreground". For our airplane recognition task, this could include things like "airport", "runway", "wings", "nose", or "wheel". However, since a machine is determining the number of clusters directly from images, there are no guarantees that topics will separate cleanly over something with specific human-understandable semantic meaning. Human interpretation of automatically generated topics from images is more difficult. For the rest of this section, we focus on explaining how machine-derived topics are arrived without respect for semantic meaning, so that we can compactly characterize an image for comparison to other images under the same topic definition (common representation).

We use Bayesian nonparametric methods as a clustering technique to compactly describe our data in terms of a number of topics (clusters) learned from the data. True to the name, *Bayesian* nonparametric methods perform *Bayesian* updates when new information is received, where it leverages a prior belief of the information explained by each input and assigns them to a topic, and updates the belief through posterior update. *Bayesian* methods are memoryless, encoding all previously processed data and associated uncertainties into its posterior belief, so we do not need to keep a record of previously processed information. In our case, we have a Bayesian probabilistic model over our clusters. Moreover, Bayesian *nonparametric* methods are special because they support flexibility of the number of clusters gleaned from the data. That is, as new information is received, the cluster definition can expand according to the properties of the data. Bayesian *nonparametric* methods are used when the number of clusters is unknown and we can add more clusters as necessary to explain our data; BNP methods differ significantly from *k-means* clustering, as the number of clusters does not need to be specified *a priori*. Instead, we learn the number of clusters (topics) from the data. By employing Bayesian nonparametric method to ingest our data and compactly characterize the data in terms of a certain set of shared descriptors, we provide a transformation from input data to a common compact representation. When refer to a common basis or common data representation, we are referring to the topic definition over shared environmental descriptors derived from HDP training described in the next section. In other words, when we compare images to other images, we do so using the topic distributions that resulted from processing each image through our HDP Experience Encoder.

Alternative to leveraging topic distributions for the foundation of our competency assessments, we could train a predictor to go directly from input data to competency predictions. However, then the approach would be limited to taking in a consistent number data streams and would not be extensible to multi-modal data stream applications when handling missing data and asynchronous data arrivals are necessary. For our obstacle recognition task, this could include simultaneous processing of inertial measurement unit (IMU), LiDAR and camera-based sensor data simultaneously. Moreover, Our BNP approach can be extended to transform observations from several sensors across disparate modalities into a common representation. And perhaps more importantly, transforming input data into a common representation provides us a way to summarize our inputs concisely and trace back our predictions (providing the *why*) to shared environmental topics. By taking this approach, we further increase the portability of our XAI capabilities to other applications and input types. And to realize the approach on our application, we implemented a hierarchical Dirichlet process (HDP) BNP approach to take in images and produce compact descriptions of the data via "topic modeling".

Section 2.1 Hierarchical Dirichlet process (HDP)

HDP is a BNP method developed in 2006 [59] that clusters data into groups based on a number of Dirichlet processes (described next). A common application of HDPs is document processing where words & "bags of words" are analyzed, and some (nonparametric) number of topics are determined using HDPs. In our case, a *document* is synonymous with an input image, and *bags of words* are synonymous with combinations of extracted image features, where each *word* is an image feature.

For example, if this dissertation document was processed, the resulting topics could include “artificial intelligence” and “machine learning”. And if you were processing many different documents, we would want the same topics to be available in the same representation to support comparisons of topic distributions across documents. We apply the same technique to our images so that we can store machine experiences compactly and infer similarity and competency information for current environments in terms of past experiences represented over a common representation of shared topic descriptors.

One benefit of HDPs is their considerable ability to compress data into compact topic distributions. Based on current sensor feeds, our HDP approach reduces storage requirements by a factor of 40,000 or more on raw sensor data and can store experiences for 1 year at less than 2 TB without implementation of additional compression or memory management techniques. This compression enables timely comparison of current environments to a vast memory of prior experiences.

At a high level, HDPs use statistical techniques to determine sufficient mixtures of features (words) and aggregations of features (topics) that explain a set of data (images). Determining the features that go with each topic and topics that represent each image is decided by Dirichlet processes. The Dirichlet process $DP(\alpha_0, G_0)$ is a distribution over probability measures; in other words, it is a distribution over distributions. It is governed by two parameters, a concentration parameter α_0 and a base probability distribution G_0. That is, for some measureable space where G_0 is a probability distribution on the space, $DP(\alpha_0, G_0)$ is the distribution of G over the measurement space so that for any finite measureable partition of the space $(A_1, A_2, \dots, A_r)$, the random vector $(G(A_1), G(A_2), \dots, G(A_r))$ is distributed as a finite-dimensional Dirichlet distribution with parameters $(\alpha_0 G_0(A_1), \dots, \alpha_0 G_0(A_r))$: $(G(A_1), \dots, G(A_r)) \sim Dir(\alpha_0 G_0(A_1), \dots, \alpha_0 G_0(A_r))$, as established by Ferguson in 1973 [59].

A common metaphor for how DPs work is represented by the "Chinese Restaurant Process" [59]. Initially, in this metaphor, some number of people are seated at some number of tables at a restaurant with an unlimited capacity on the amount of tables that can fit in the restaurant. A new guest arrives and sits down at an existing table proportional to the number of people sitting at the table or at a new table proportional to the concentration parameter α, set *a priori*. Specifically, a new customer sits at an existing table with probability given in Eq. 17, where n_k represents the number of people currently seated at table *k* and *(i-1)* is the total number of customers already seated at the restaurant when the new customer arrives. Meanwhile, the probability of sitting at a new table is given in Eq. 18. We note that this promotes a "rich get richer" scheme, and that a new customer is more likely to sit at an occupied table if many people are seated. This is known as the "clustering effect", introduced along with the Chinese restaurant process in [60]. We also note that the greater the concentration parameter α, the more likely a new table is to form.

$$Probability\ of\ sitting\ at\ existing\ table\ k = \frac{n_k}{\alpha + i - 1} \quad (Eq. 17)$$

$$Probability\ of\ sitting\ at\ new\ table = \frac{\alpha}{\alpha + i - 1} \quad (Eq. 18)$$

The HDP extension of this DP metaphor is a Chinese Restaurant Franchise process, wherein the HDP determines the number of restaurants, the number of tables in each restaurant, and the number of shared dishes across the franchise. In our application, this is analogous to the number of topics, the composition of features present within each topic, and the number of environmental feature descriptors shared across topics, respectively.

Integral to the HDP approach is the definition of a base distribution that is itself a draw from a Dirichlet process $DP(\gamma, \beta)$. This is how we instrument hierarchical links between a global topic distribution, governed by scaling parameter γ and base probability measure β over a local topic distribution, where a Dirichlet process for each topic is governed by scaling parameter α_0 and a base probability distribution G_0. For our application, the local topic definition refers to the combinations of features that make up a specific topic. The global topic definition refers to how many topics we have overall. In summary, the HDP approach is specified by Eq. 19 and Eq. 20 [59] and illustrated in Figure 32 [45].

$$G_0 | \, \gamma, \beta \sim DP(\gamma, \beta) \qquad \text{(Eq. 19)}$$

$$G_m | \, \alpha_0, G_0 \sim DP(\alpha_0, G_0) \; for\ each\ topic\ m \qquad (Eq. 20)$$

In simplest terms, α_0 determines when a new feature is assigned to a topic. And γ determines when a new topic is necessary to describe the data that has been processed. The complexity of the model adapts to the composition of features within each environment as well as the number of distinct environments. Both concentration parameters (α_0 and γ) are tunable and can give rise to more or fewer clusters, accordingly. We performed a parametric study to determine the best choices for the concentration parameters using MIT’s high-performance computing environment, and the results of the study are omitted from this dissertation.

For our application, we process images (documents) into compact descriptions (topics) of machine experiences using HDP. The topic definitions are learned offline using the method described. However, for online use, we freeze the number and feature composition of topics and infer topic proportions of new images processed. This is so we can compare new data to previous machine experiences under a common representation. During offline training, the HDP learns topic distributions that best characterize the data, allowing topic definitions to fluctuate. Online, we make inferences over new data under the trained representation. In this way, we guarantee that references to specific topics retain the same meaning over both offline and online phases.

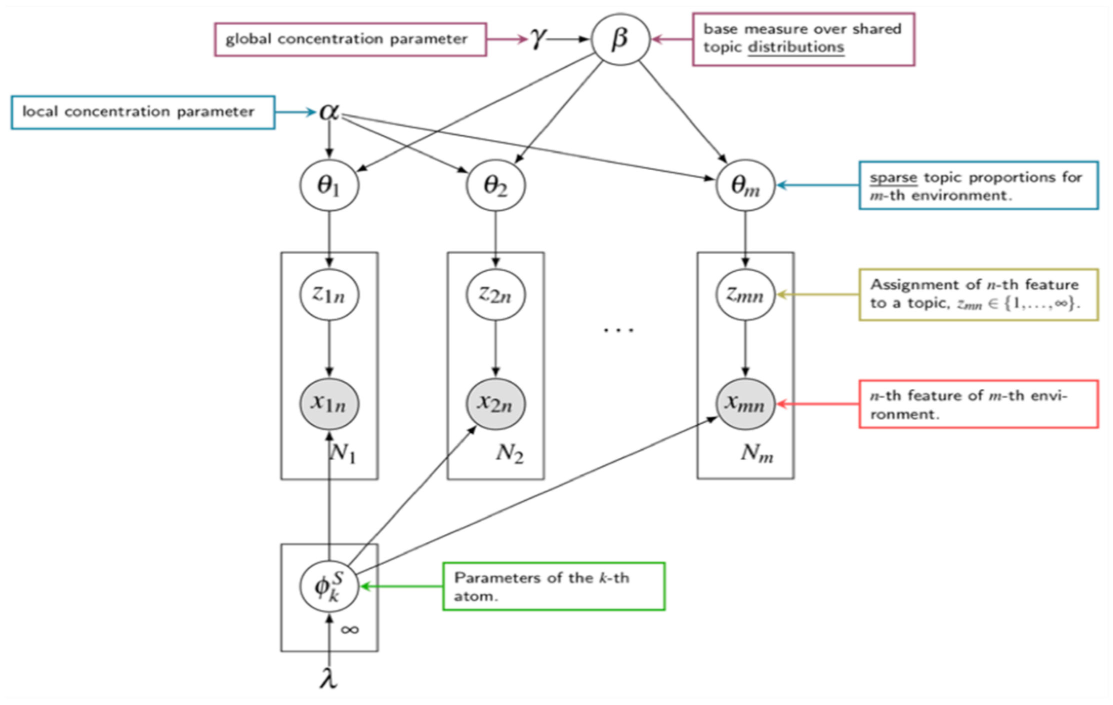

global concentration parameter
base measure over shared topic distributions
local concentration parameter
sparse topic proportions for m-th environment.
Assignment of n-th feature to a topic, $z_{mn} \in \{1,\ldots,\infty\}$.
n-th feature of m-th environment.
Parameters of the k-th atom.
γ
β
α
θ_1
θ_2
θ_m
z_{1n}
z_{2n}
z_{mn}
x_{1n}
x_{2n}
x_{mn}
N_1
N_2
N_m
ϕ_k^S
∞
λ

Figure 32 HDP of observations into compact representations and traceable conditions [45]

In Figure 32, we show the HDP model. Features (x_{mn}) are extracted from the input image (ϕ_k^S) and compared to the feature represented associated with each topic

(z_{mn}), which were governed by the concentration parameter α during training. The incoming document is represented by ϕ_k^S from some sensor S. Data are organized into separate groups each containing unstructured data features (x_{mn}). All features are derived from a common data type across all documents and combinations of features are associated with expressions of each topic. New images are aligned with the set of topics learned from previous experience. During training, concentration parameter γ determines when a new topic is necessary to sufficiently capture the features expressed in the data. **The resulting histogram of topic distributions is the compact representation of the machine experience that is saved into memory and lays the foundation for competency predictor components and environment similarity calculations.** The image (environment) associated with the current task is thus represented compactly by topic proportions determined by a trained HDP model. And each previously experienced environment is described as a unique symbolic mixture over the same set of shared topics.

The use of HDPs addresses several critical issues. HDPs are models over shared descriptors; different environments are described by distinct proportions of topics. Additionally, by design, each individual environment is a sparse mixture, meaning only a few topics are present for each environment. The result is a compact representation that explicitly encodes shared properties of different environments in the global topic definition. We use the same HDP techniques to encode the current operating environment into a local topic definition, thereby identifying emergent environmental conditions automatically and providing the basis for making competency predictions traceable back to topic dependencies (conditions).

Analysis and comparison of topic distributions is used to predict task competency. With reference to Figure 32, topic proportions (θ) are a compact sufficient statistic of historical data sets and experience [56] [57], resulting in a dramatic reduction in required storage capacity. Consequently, reasoning over novel environments is with respect to these parameters. The application of HDPs for enabling competency predictions is novel, expanding on prior research using HDPs to craft compact data representations [56] [57].

Over our images for the obstacle recognition task, HDPs work on features (words) that are extracted from the image (document). While a multitude of feature extraction methods can be used, we chose to use Oriented FAST and Rotated BRIEF (ORB) feature extraction. ORB feature extraction is available for free from OpenCV and describes local features in images [61]. The pre-processing pipeline for transforming images into features for inputs to the HDP is involved, including feature extraction, summarization, and quantization, for which details are omitted in this dissertation.

The procedure illustrated in Figure 33 helps to summarize the HDP approach used as the foundation for competency prediction in this chapter. An image is input, features are extracted (top) and associated with topics (right) according to the feature-topic memberships. Finally, topic proportions are determined (bottom) based on the feature-to-topic analysis (right) that compactly describe the image. Every image is processed this way.

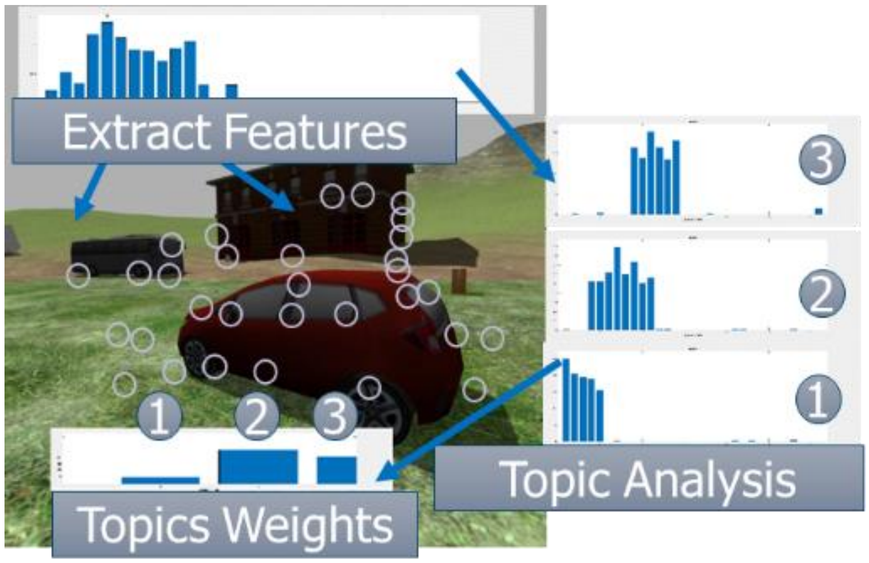


Figure 33 HDP procedure illustration

In the following results, we used an HDP as described above and used ORB feature extraction with additional spatial context. That is, we delineated extracted features based on their location in the image. An example of this "ORB + spatial context" feature extraction is illustrated in Figure 34. In the top left of the figure, we can see the red, orange, dark blue, light green, and light blue spatial bins that were defined *a priori*. Then, within those bins, ORB features are extracted, where different orb features correspond to different color intensities. On the top right, we see the magnitude of HDP features (combinations of ORB features shown on the y-axis) represented by color intensity extracted from each image document (x-axis). In the bottom left, we show the original image, as collected from the camera sensor in the ROS/Gazebo simulation. And in the bottom right, we show the HDP-derived distribution of topics over time, where the document index (image frame index) is on the x-axis, topic weights are on the y-axis, and the color represents varying topic indices. Both the top right and bottom right plots show results for a large set of images. The example images on the left correspond to the first image index. We can play the images as a video over time to conceptually show how the features and topic expressions evolve over time. A snippet of the top row of frames is available online at https://www.linkedin.com/feed/update/urn:li:activity:6752672510229143552/, playing in the opening scene of the video. As the documents are ordered by frame within a certain environment, it is encouraging that we see homogeneity in the topic expressions in certain regions of orange (Star 1) and light blue (Star 2). This is an important result that supports strong qualitative attribute of "stability" discussed previously. Otherwise, if topic distributions varied significantly from frame to frame, we would feel less confident that we could fairly compare the current environment to past experiences adequately.

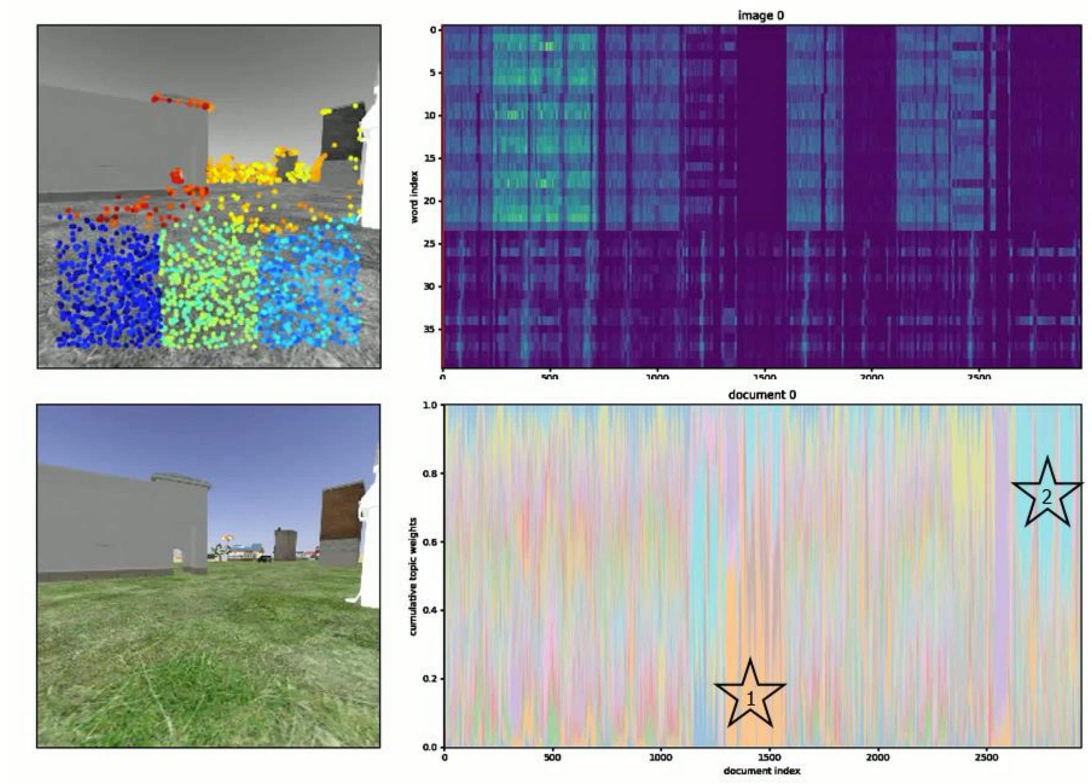


Figure 34 Illustration of ORB feature extraction with spatial context and associated topic analysis

The measure we used to calculate topic distributions similarity is a Kullback-Leibler divergence (KL-divergence) score (Eq. 21). KL-divergence measures the difference between two probability distributions, and is thus a reasonable measure to use to assess the similarity between one topic distribution and another.

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \quad (Eq. 21)$$

Here, Q(x) is the distribution over the weights x of HDP feature expression based on topic proportions from a past environment; $P(x)$ is the distribution over the weights x of feature expression proportions from the current environment. This measure is also foundation for our Environment Similarity Calculator, described in Section 3.

Next, we attempt to qualitatively assess HDP processing of sensory inputs into topic distributions. We introduce "stability" and "consistency" as two <u>qualitative</u> attributes of HDP performance; we do not make any claims about significance or quantitative assessments that in any way prove that we have adequately captured important characteristics of the image for determining machine competency. However, both sets of results are encouraging qualitatively and indicative of "stability" and "consistency".

First, "stability" refers to the sensitivity of topic distributions from frame to frame. We would like the topic distributions to be stable in the sense that they vary smoothly when the changes in the image are small. Second, "consistency" refers to the topic distribution variation over different environmental properties. We would like the topic distributions to follow a similar pattern when the ground robot passes through environments where only an environmental property varies, such as snow versus rain. We attempt to qualitatively assess both of these desirable attributes next.

To assess HDP stability, we calculated the KL-divergence score from one image to the next image for two ground robot paths taken through the ROS/Gazebo simulation as shown in Figure 35. And, as we had hoped, images with similar visual content have reduced KL-divergence (y-axis) when assessed over time (x-axis). From frame to frame, KL divergence varies smoothly when inspected by eye, indicating HDP model is not sensitive to small changes in the environment.

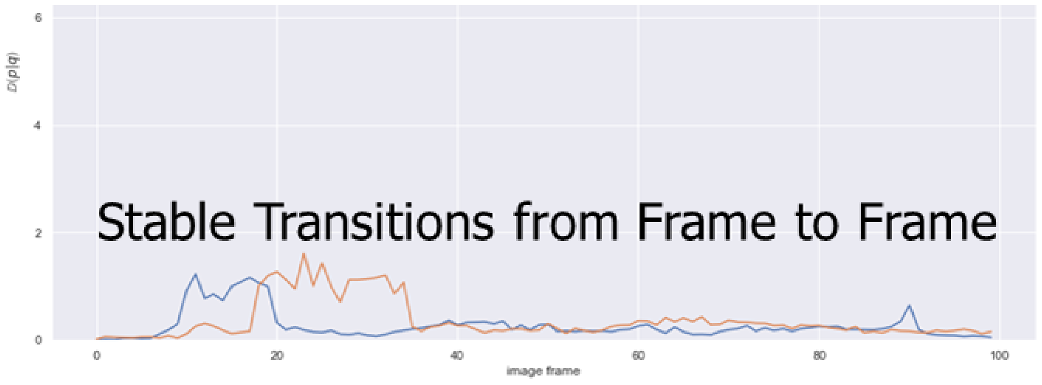


Figure 35 HDPs produce stable transitions from frame to frame

To assess HDP consistency, we collected data across 10 simulated environmental properties, as illustrated in Figure 36. We hope to see similar topic distributions associated with each environmental property. And we also hope to see similar patterns in topic distributions as the robot took the same approximate path through each of the 10 environments.

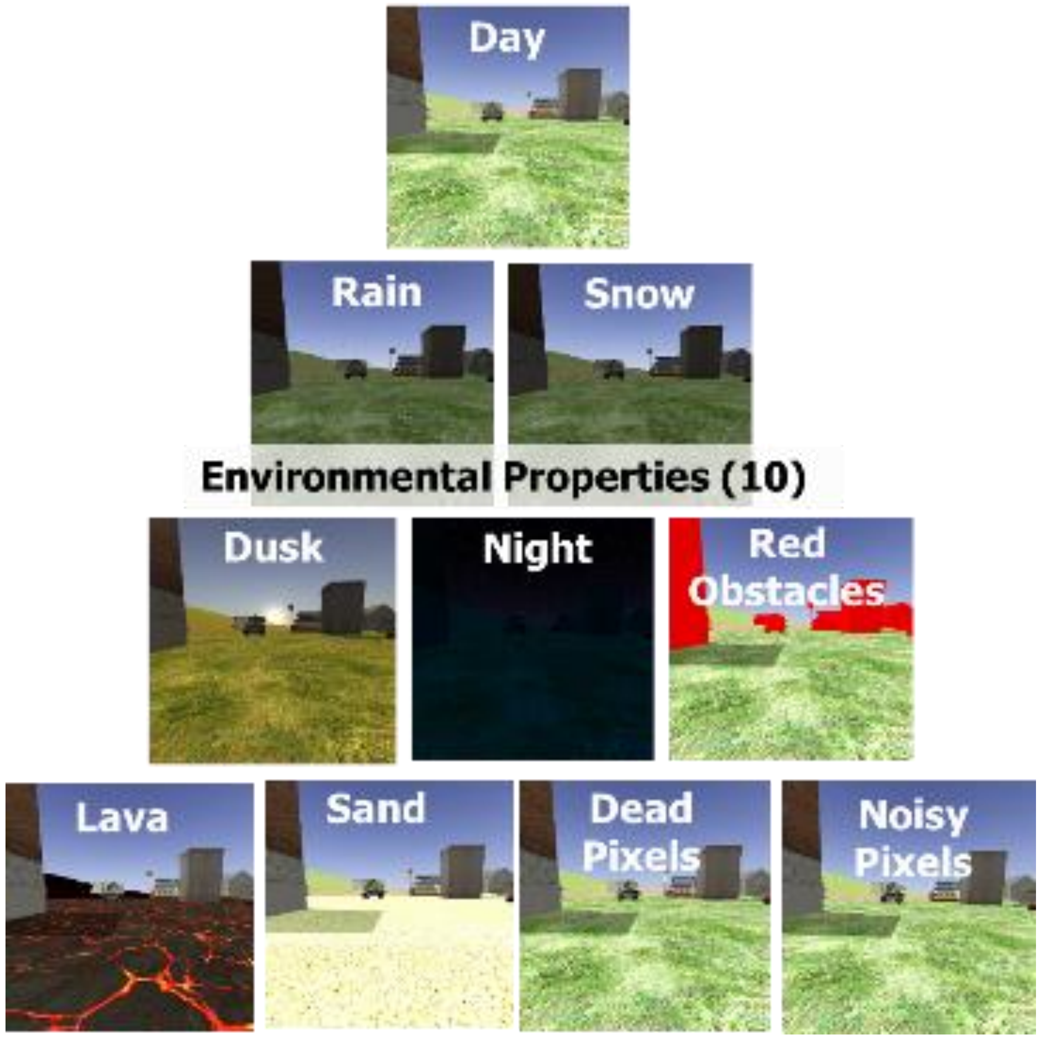


Figure 36 Data collected over 10 different simulated environmental properties

In Figure 37, we show the KL-scores (y-axis) for each environment (white gridlines) over time (x-axis). We also show how KL-scoring for each environment compares to each other environment. Each of the gridlines separates data collected from each environment given in Figure 36 and how it compares to data collected from another environment. Intuitively, the results make sense. When we perform visual inspection, for example, we see that the center column (night environmental property), is very different (high KL-score, lighter heat index) than other most other environments (except itself). Moreover, we see a repeated cross pattern within each grid square, showing that the robot path through each environment is experiencing similar fluctuation across each property. This indicates, qualitatively, that the HDP model is consistent across environmental properties, and we have confidence that our HDP approach provides stable topic distributions from small changes in the frame and produces comparable topic distributions from environmental property to property.

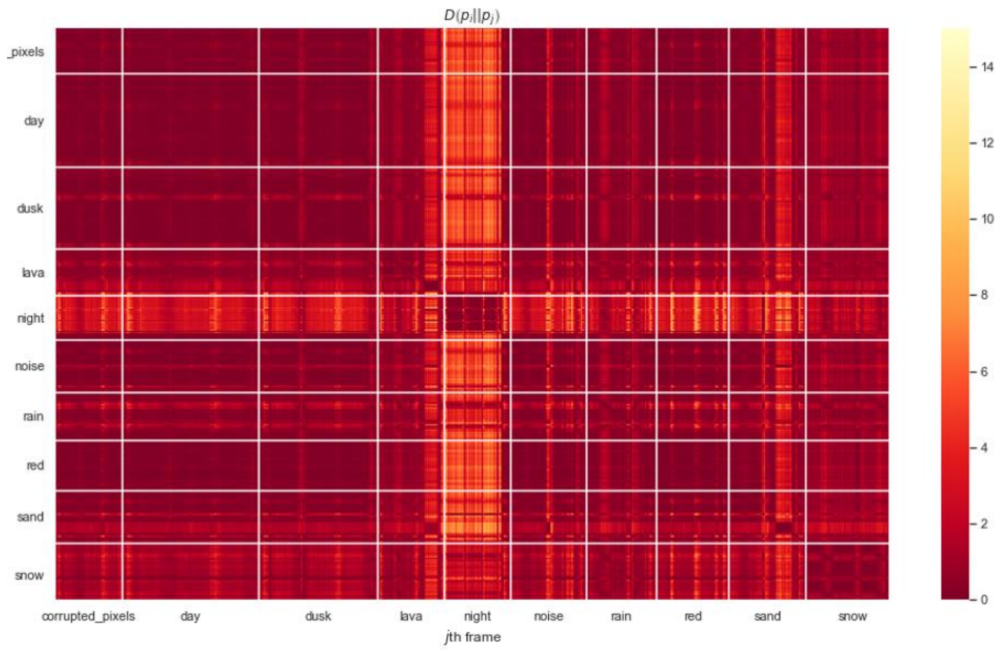


Figure 37 HDP approach yields consistent results within each property and varies intuitively across properties

Section 2.2 Semantic Interpretability of Topics

While human interpretability of HDP-derived topics is still an ongoing research area, preliminary results support that HDPs can pick up on environmental descriptors that have semantic meaning. When we refer to "semantic meaning", we refer to word labels associated with HDP-derived topics that a human can understand, such as "airport", "wings", "nose", and "wheel" for our airplane imagery exploitation task. For our obstacle recognition task, when we simulate an environment with a lava ground appearance and input and interpret labels that are available automatically through Alexnet processing, we notice that images with high expression of Topic 9 (Figure 38) are associated with the label "volcano", as provided in the MEIngester UI (Figure 39).

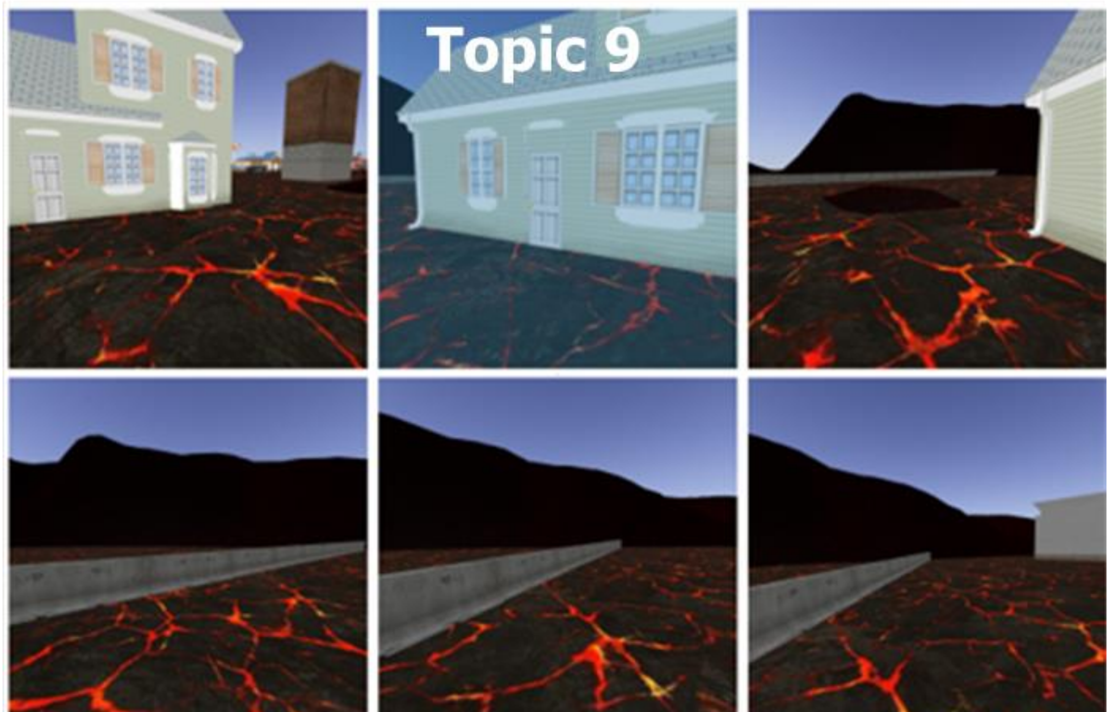
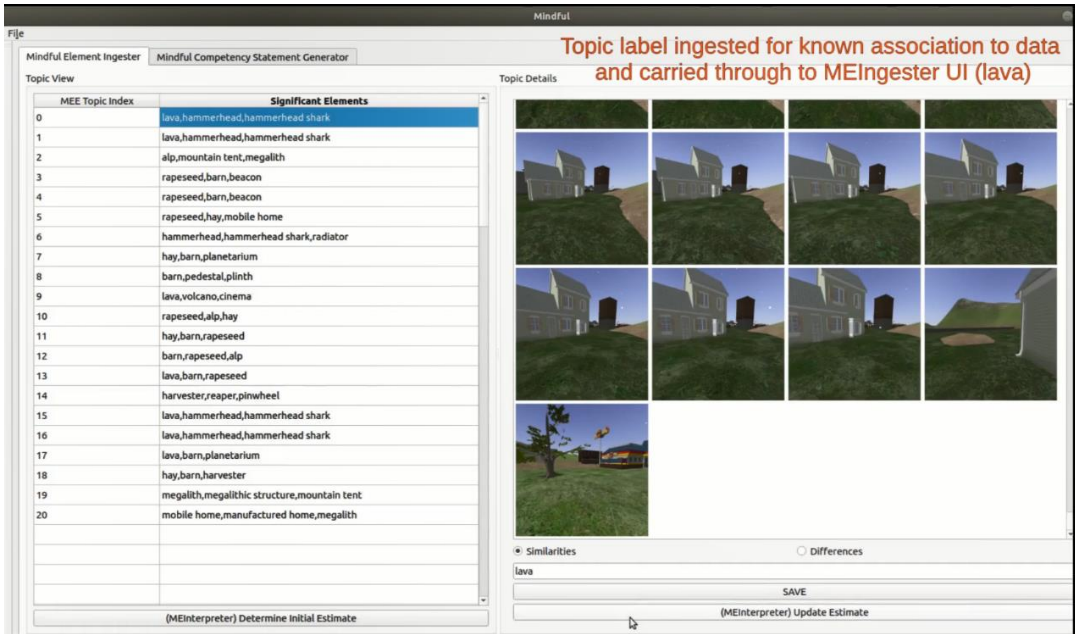


Figure 38 Alexnet machine-provided labels & the MindfuL™ Element Interpreter automatically attach the word "volcano" to Topic 9, which is often highly expressed in the lava simulation environment

While not described in detail in this dissertation, the MindfuL™ team developed an interface, called the Element Ingester (Figure 39), which supports efficient labeling efforts, where labels are interpreted and propagated through to topic label estimates via a Bayesian update method inherent to the Element Interpreter component.


Mindful
File
Mindful Element Ingester
Mindful Competency Statement Generator
Topic label ingested for known association to data and carried through to MEIngester UI (lava)
Topic View
Topic Details
Similarities
Differences
lava
SAVE
(MEInterpreter) Determine Initial Estimate
(MEInterpreter) Update Estimate

MEE Topic Index	Significant Elements
0	lava,hammerhead,hammerhead shark
1	lava,hammerhead,hammerhead shark
2	alp,mountain tent,megalith
3	rapeseed,barn,beacon
4	rapeseed,barn,beacon
5	rapeseed,hay,mobile home
6	hammerhead,hammerhead shark,radiator
7	hay,barn,planetarium
8	barn,pedestal,plinth
9	lava,volcano,cinema
10	rapeseed,alp,hay
11	hay,barn,rapeseed
12	barn,rapeseed,alp
13	lava,barn,rapeseed
14	harvester,reaper,pinwheel
15	lava,hammerhead,hammerhead shark
16	lava,hammerhead,hammerhead shark
17	lava,barn,planetarium
18	hay,barn,harvester
19	megalith,megalithic structure,mountain tent
20	mobile home,manufactured home,megalith

Figure 39 Additional MindfuL™ Element Ingester interface supports human labeling.

Moreover, we spent approximately 4.5 person-hours to augment the automatically-available Alexnet label set to improve topic semantic interpretability by annotating data conditions that an expert proposes might have effects on underlying competency. The set of expert-proposed condition labels is provided in Table 12.

Table 12 Expert-Proposed Condition Labels

Expert-Proposed Condition Labels
Object off to left side
Object off to right side
Clear (when no obstacle is present and foreground is clear)
Dark
Shadow
Building close
Barrier (for the jersey barrier/wall)
Obstacle very close
Obstacle close
Smooth surface (this goes with the red objects)

Using the Element Ingester, the annotator labeled images that seemed to exhibit any of the expert-proposed condition labels. The Element Ingester interface supported mass data labeling and the Memory Bank database stored relationships between labels and data persistently. As a result of the labeling effort, our topics had more human-relatable context. This is important for determining and communicating environmental conditions we suspect strongly influence our strategy and performance predictions, described in Section 7. Example labels, associated data, and high topic expressions are shown in Figure 40.

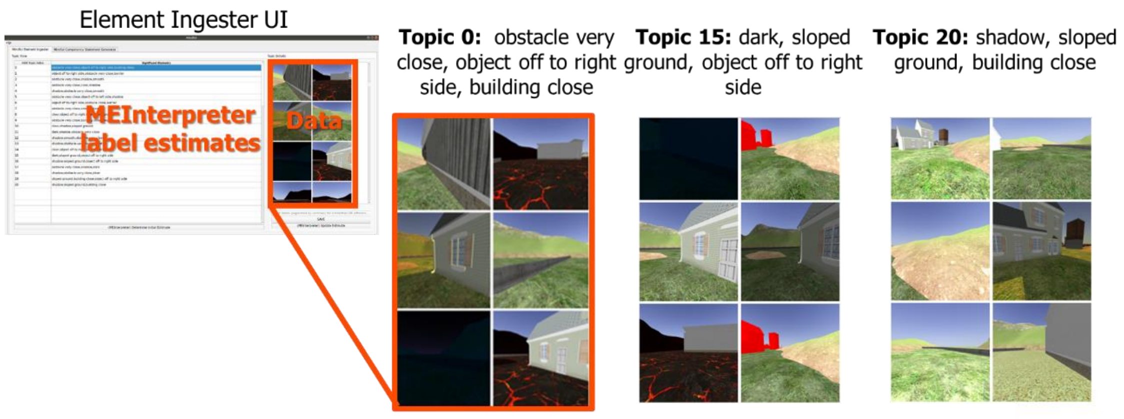

Element Ingester UI
MEInterpreter label estimates
Data
Topic 0: obstacle very close, object off to right side, building close
Topic 15: dark, sloped ground, object off to right side
Topic 20: shadow, sloped ground, building close

Figure 40 Element Ingester & Interpreter components carry human-provided labels through to improve topic semantic interpretability

Section 3: Environment Similarity Calculator

3.1 Assessing current environment similarity to past machine experiences using KL-divergence scoring

Since the HDP experience encoder compactly represents ingested data in a common representation via topic distributions, we can perform similarity calculations over all prior experiences by comparing the topic distribution associated with the current environment with those of past experiences. Similarity between the current environment and historical environments, as defined as the similarity between two topic distributions, increases confidence in ML capabilities in the current environment, accuracy of Mindful performance predictions, and accuracy of behavior-controlling topics.

We capture the number of experiences succinctly by performing pairwise KL-divergence (Eq. 21) calculations over the current topic distribution and all previously stored topic distributions (machine training experiences). In Figure 41, we show the results of comparing the current document to all prior experiences, as encoded by their topic distributions. The top row represents the most similar past experiences and the bottom row represents the least similar past experiences, as determined by KL-scoring. Through manual inspection, we see that the top row contains images that look similar to the human eye. Conversely, the bottom row represents the least similar images, which also makes intuitive sense.

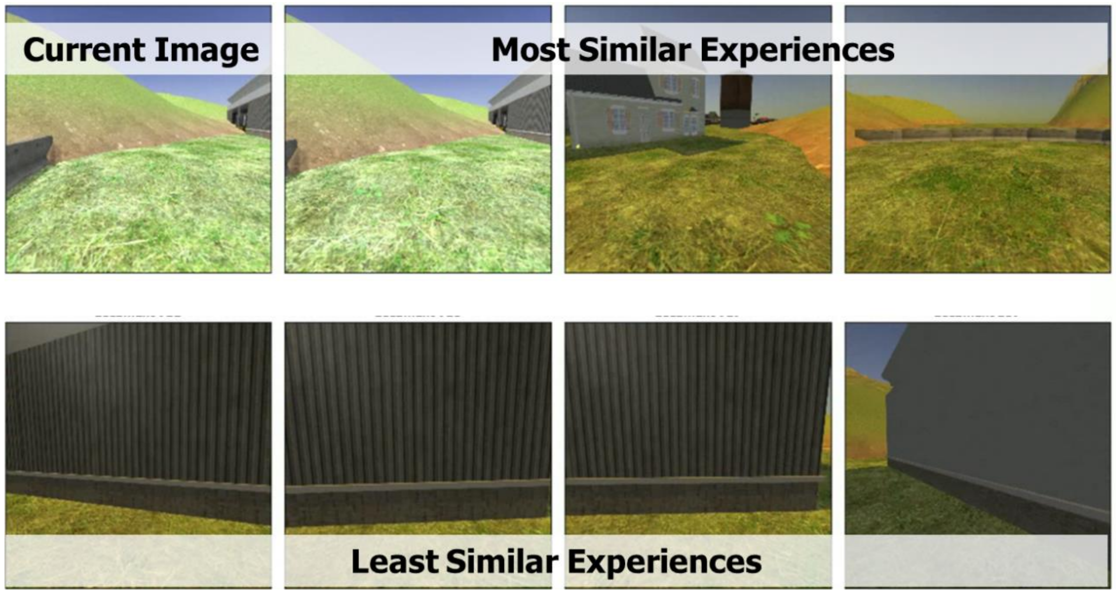


Figure 41 Environment Similarity Calculator results comparing a current image to images in the training data set. The top left image is the image being tested for similarity to previous experiences. The rest of the top row are the most similar images and the bottom row are the least similar images, as characterized by their pairwise KL-divergence score.

As the HDP offers considerable compression benefits, we can perform pairwise calculations of current images over past experiences very quickly. In fact, performing all of the competency predictions, similarity calculations, natural language sentence generation, and user interface population takes less than 2 seconds over 10,000 past experiences, stored compactly in the Memory Bank. We note that this could be sped up considerably by parallelizing the similarity computations, however, the resulting update rate is faster than human preference, so there are no planned efforts for parallelization on this proof-of-concept application.

3.2 Historical competency distributions filtered by similar past experiences

Another advantage of categorizing whether a past experience is similar to the current environment is the ability to characterize historical competency in similar environments. To enable this analysis, in addition to storing topic distributions associated with training documents, we store the metadata associated with how and how well the machine performed its task in that environment. In other words, we store the historical performance (whether the machine performed its task well) and historical strategy (as defined in Section 4.1) along with the topic distribution in the Memory Bank.

As such, for the set of similar documents determined by the Environment Similarity Calculator, we can provide a breakdown of the historical performance distribution and historical strategy distribution. And if we have an online strategy prediction for the current environment (as derived in Section 5) we can further break down historical performance for each strategy class prediction. This concept is illustrated in Figure 42.

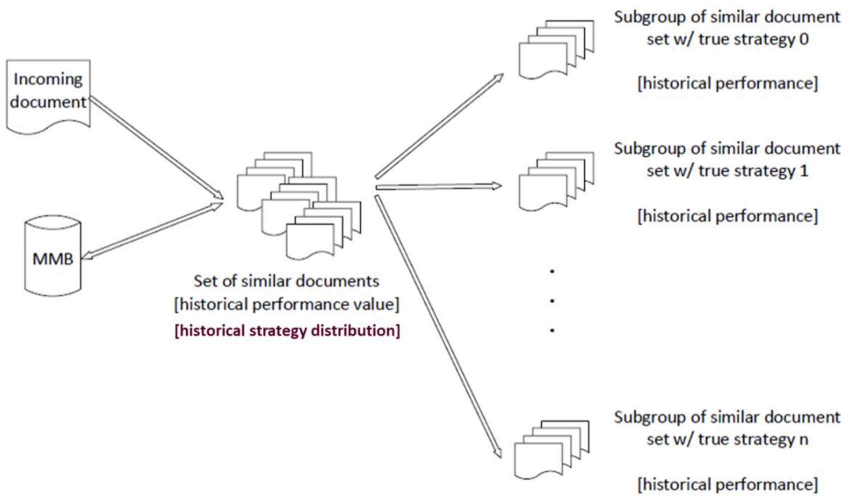

Incoming document
MMB
Set of similar documents
[historical performance value]
[historical strategy distribution]
Subgroup of similar document set w/ true strategy 0
[historical performance]
Subgroup of similar document set w/ true strategy 1
[historical performance]
Subgroup of similar document set w/ true strategy n
[historical performance]

Figure 42 Providing historical competency information through similar experience filtering

3.3 Leveraging environment similarity to detect data ingestion anomalies and sensor faults

Lastly, another redeeming property of assessing the similarity of the current environment to training experiences is the detection of data anomalies. That is, we can determine when data inputs look significantly different (anomalous) compared to prior data inputs. Not only can we identify when the ML system has not been trained on an environment similar to the current one, but we can also detect when a different type of data is being ingested, as it is classified as highly dissimilar (anomalous) compared to previously ingested data. This can thus be used to detect when a sensor is not acting as it has previously and other data anomalies. We tested this hypothesis by simulating an event where the camera fell out of the housing on the ground robot used in the ROS/Gazebo simulation to collect data. As desired, the Environment Similarity Calculator outputs responded accordingly. Specifically, the number of similar experiences dropped drastically from ~10,000 to ~200 for the same image taken with a fully functional sensor and broken sensor. Thus, our environment similarity calculator can detect an unfamiliar environment (anomalous sensor data) when a sensor fault is simulated.

Section 4: Offline Definition of ML Strategies & Performance

4.1 Offline Definition of ML Strategies

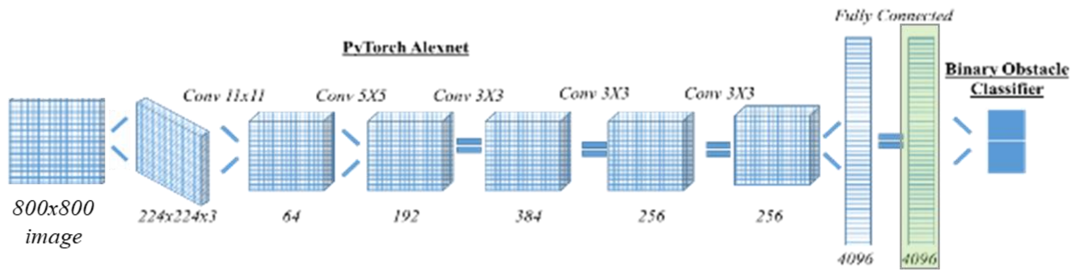


Figure 43 Behavior Definition Network View. Outputs from the 4096 nodes in the second-to-last layer (highlighted in green) are used as our behavior definition.

We used the same method (Table 10) in Chapter 3 to define a set of machine strategies. In this case, we defined our behavior to be the outputs (y_i outputs from each i^{th} node, defined in Figure 14), from the 4096 nodes in the second-to-last layer of the ML system under evaluation (highlighted in green in Figure 43). Note that this is analogous to the 702-dimension behavior definition from Chapter 3. However, as the ML system performs its task on a per-instance basis, no time-series behaviors are defined.

Additionally, we employ the same elbow analysis technique as we did in Chapter 2. Here, we find that 11 clusters gives us both an acceptable inertia score and the right granularity of strategies (Figure 44). For the latter, we manually examined data associated with each of the 11 strategies and found that 9 of them had human-interpretable meaning.

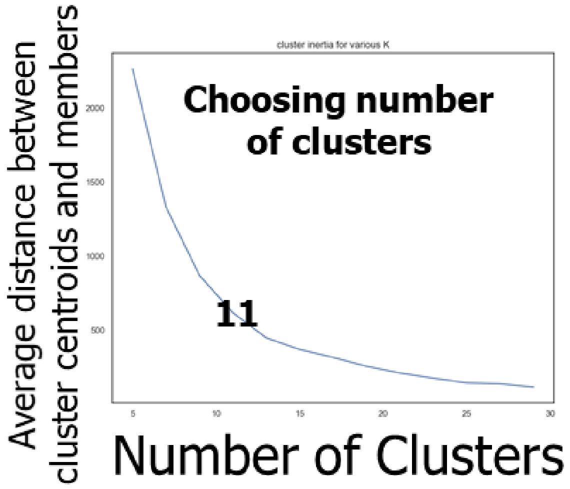


Figure 44 Elbow method analysis to inform number of strategies

Figure 45 shows the strategies that resulted from the k-means analysis. In this case, one strategy stands out from the rest. Upon manual inspection, we see that all images associated with that cluster have very low feature density and correspond to the "night" environmental property.

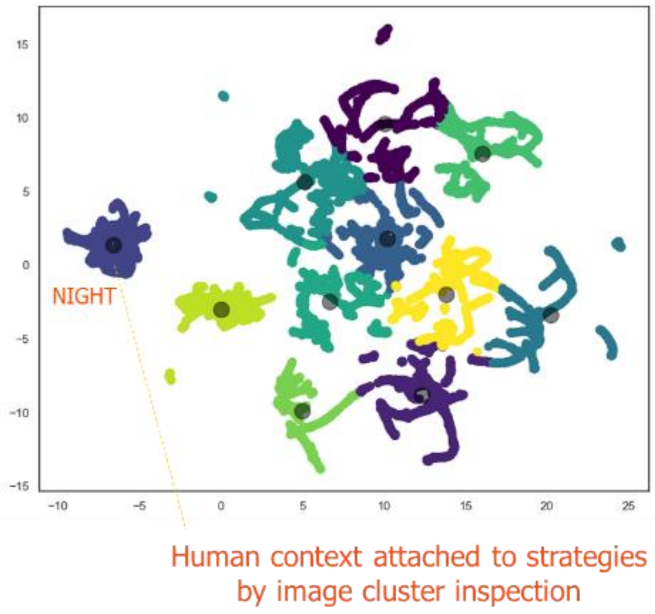


Figure 45 UMAP Visualization of Strategy Clusters in an abstract two-dimensional space where only relative distances between points is interpretable.

Additionally, 8 other clusters had human-interpretable meaning and are tagged in Figure 46. Since strategy groupings are frozen after initial offline determination, we can do this manual inspection analysis just once. The labels associated with each strategy offline are then available for association to online strategy predictions of the same name.

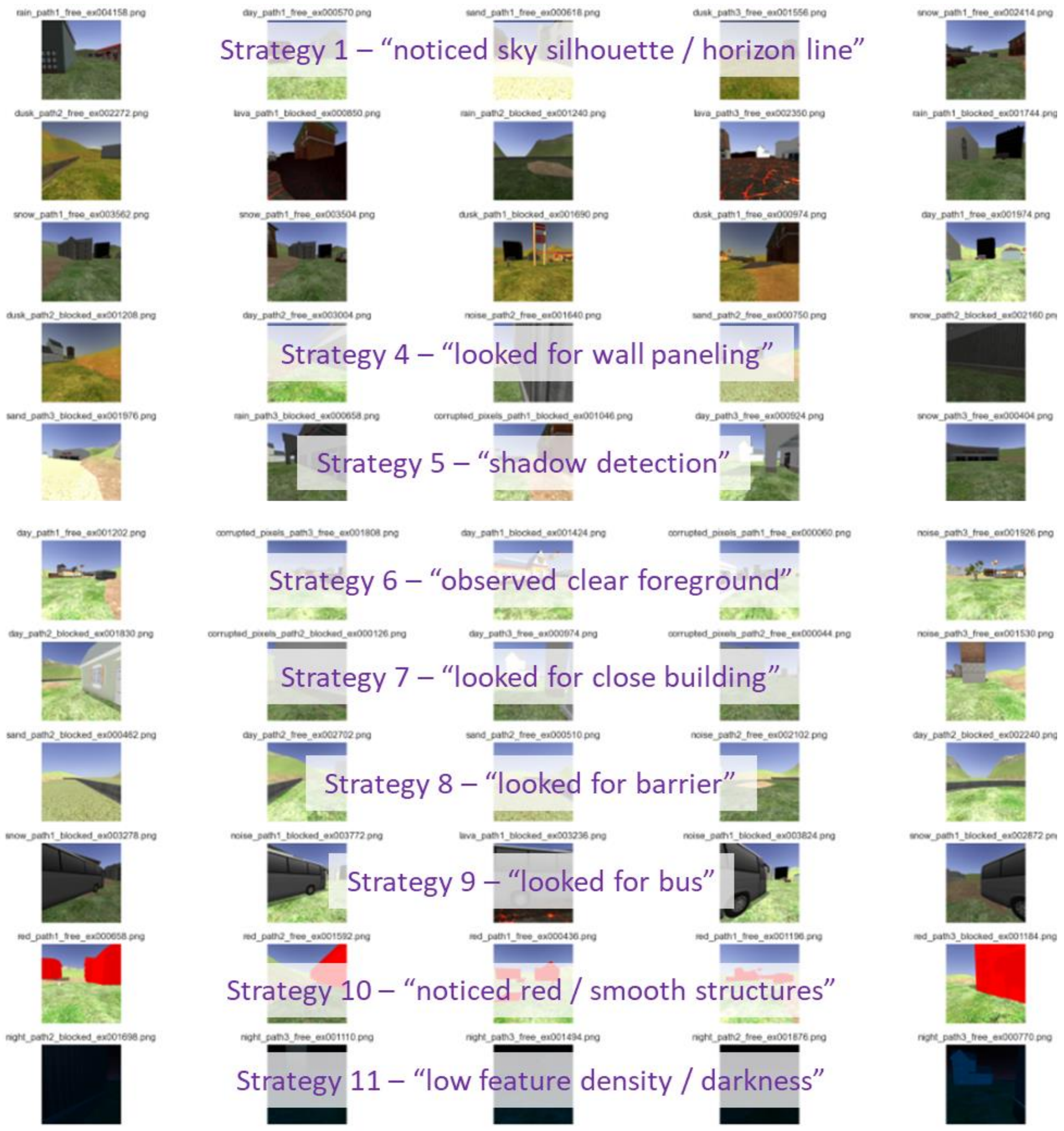
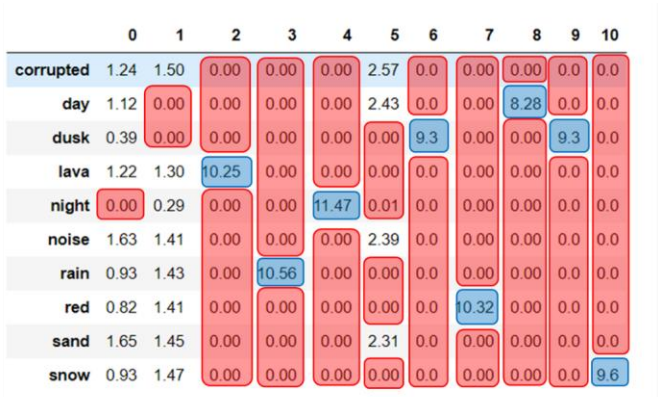

Strategy 1 – "noticed sky silhouette / horizon line"
Strategy 4 – "looked for wall paneling"
Strategy 5 – "shadow detection"
Strategy 6 – "observed clear foreground"
Strategy 7 – "looked for close building"
Strategy 8 – "looked for barrier"
Strategy 9 – "looked for bus"
Strategy 10 – "noticed red / smooth structures"
Strategy 11 – "low feature density / darkness"

Figure 46 9 of 11 Strategies (clusters) have human-interpretable meaning!

Some strategies do not have human-interpretable meaning. In these cases, the ML system is picking up on groups of features that are not easily categorized by the human eye, or that require further separation (deeper granularity) to become interpretable. We experimented with several numbers of clusters, as we did in Chapter 3, however, $k=11$ resulted in the most human-interpretable context across all clusters.

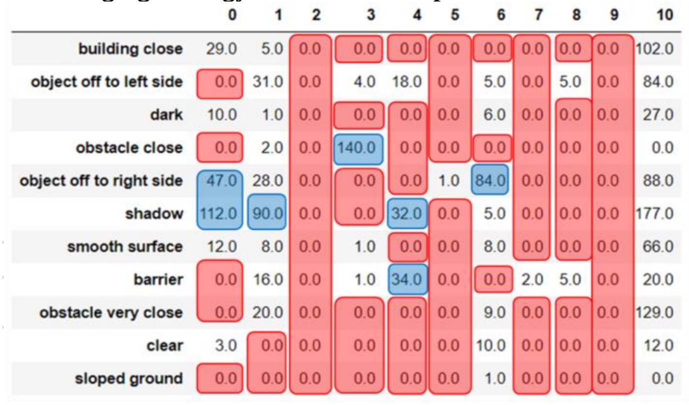
While behavior clustering revealed some human-understandable context, we are experimenting with incorporating separation over environmental properties and expert-proposed condition labels to achieve semantic separation amongst strategies. To do this, we adjust the cluster "gain" ratio in the k-means algorithm to nudge separation of strategies from one another over a given set of labeled categories. This encourages homogeneous organization of strategies over desired properties. The following discussion and figures shares preliminary results associated with this line of research, however, effects on downstream competency assessment performance have not yet been evaluated.

Figure 47 and Figure 48 show the strategy distributions resulting from these adjustments. In each figure, we show Strategy clusters across the top, labels along the left hand side, and the ratio of the label in the set of data associated with each strategy cluster. In this case, values close to 1 indicate that the strategy ratio is consistent with the population proportion. In other words, values greater than 1 indicate that there are more of that label present in the associated strategy than the population as a whole. For example, Strategy 2 has 10.25x more "lava" labels than the population. We manually highlight the cases that have higher expression than normal in blue.



	0	1	2	3	4	5	6	7	8	9	10
corrupted	1.24	1.50	0.00	0.00	0.00	2.57	0.0	0.00	0.00	0.0	0.0
day	1.12	0.00	0.00	0.00	0.00	2.43	0.0	0.00	8.28	0.0	0.0
dusk	0.39	0.00	0.00	0.00	0.00	0.00	9.3	0.00	0.00	9.3	0.0
lava	1.22	1.30	10.25	0.00	0.00	0.00	0.0	0.00	0.00	0.0	0.0
night	0.00	0.29	0.00	0.00	11.47	0.01	0.0	0.00	0.00	0.0	0.0
noise	1.63	1.41	0.00	0.00	0.00	2.39	0.0	0.00	0.00	0.0	0.0
rain	0.93	1.43	0.00	10.56	0.00	0.00	0.0	0.00	0.00	0.0	0.0
red	0.82	1.41	0.00	0.00	0.00	0.00	0.0	10.32	0.00	0.0	0.0
sand	1.65	1.45	0.00	0.00	0.00	2.31	0.0	0.00	0.00	0.0	0.0
snow	0.93	1.47	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.0	9.6

Figure 47 Encouraging strategy definitions to separate over environmental properties



	0	1	2	3	4	5	6	7	8	9	10
building close	29.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	102.0
object off to left side	0.0	31.0	0.0	4.0	18.0	0.0	5.0	0.0	5.0	0.0	84.0
dark	10.0	1.0	0.0	0.0	0.0	0.0	6.0	0.0	0.0	0.0	27.0
obstacle close	0.0	2.0	0.0	140.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
object off to right side	47.0	28.0	0.0	0.0	0.0	1.0	84.0	0.0	0.0	0.0	88.0
shadow	112.0	90.0	0.0	0.0	32.0	0.0	5.0	0.0	0.0	0.0	177.0
smooth surface	12.0	8.0	0.0	1.0	0.0	0.0	8.0	0.0	0.0	0.0	66.0
barrier	0.0	16.0	0.0	1.0	34.0	0.0	0.0	2.0	5.0	0.0	20.0
obstacle very close	0.0	20.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	129.0
clear	3.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	12.0
sloped ground	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

Figure 48 Encouraging strategy definitions to separate over expert-proposed conditions

By continuing this line of research, we contribute to the semantic interpretations of strategies produced by the MindfuL™ system, an important aspect of the DARPA CAML program and XAI research.

4.2 Offline definition of ML Performance

Offline definition of ML Performance is much simpler. For the obstacle recognition task, we define a binary performance measure that takes on values of “correct” or “incorrect”, based on whether the ML correctly recognized the presence of an obstacle within six meters of its position.

Section 5: Online Prediction of ML Strategies

We use a deep learning approach to predict strategies from topic distributions. In this case, we train on topic distribution inputs and learn strategy classifications in accordance with the UMAP representation. That is, each training sample in Figure 45 is associated with an image. The image gets compressed by the HDP into a topic distribution. And our Strategy Predictor is trained to learn transformations from topic distributions into strategy classifications. The deep learning neural network structure for our strategy predictor approach is provided in Figure 49.

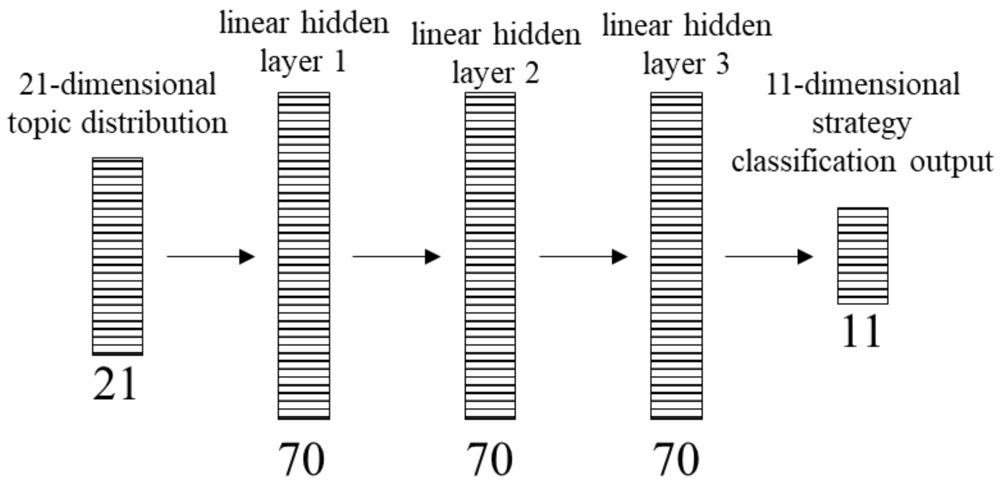

21-dimensional topic distribution
21
linear hidden layer 1
70
linear hidden layer 2
70
linear hidden layer 3
70
11-dimensional strategy classification output
11

Figure 49 Strategy Predictor Network Structure, where "hidden" refers to the deep network layers between the inputs and the outputs.

In our preliminary approach, we achieved 64% correctness (Eq. 22) using this framework. That is, for 48,000 images, MindfuL™ correctly predicted 30,561 strategies correctly.

$$\text{Correctness} = \frac{\text{Number of times system predicted the correct } strategy}{\text{Total number of trials}} \quad (Eq.22)$$

The associated receiver operating characteristic (ROC) curves for each Strategy Class are shown in Figure 50. The ROC curve shows how the true positive and false positive rates vary as a function of setting different ML probability thresholds. As the threshold for “ML probability blocked” corresponding with a binary “blocked” classification gets looser (lower), the True Positive Rate and False Positive Rate increase. That is, if we give the ML credit for correctly predicting that an obstacle was present based on a low threshold confidence estimate (such as 0.1 probability “blocked”), the resulting True Positive Rate and False Positive Rate are higher. The greater the area under the curve, the better the predictor. In the figure, each class corresponds a different strategy. And we see that some strategies are easier to predict than others.

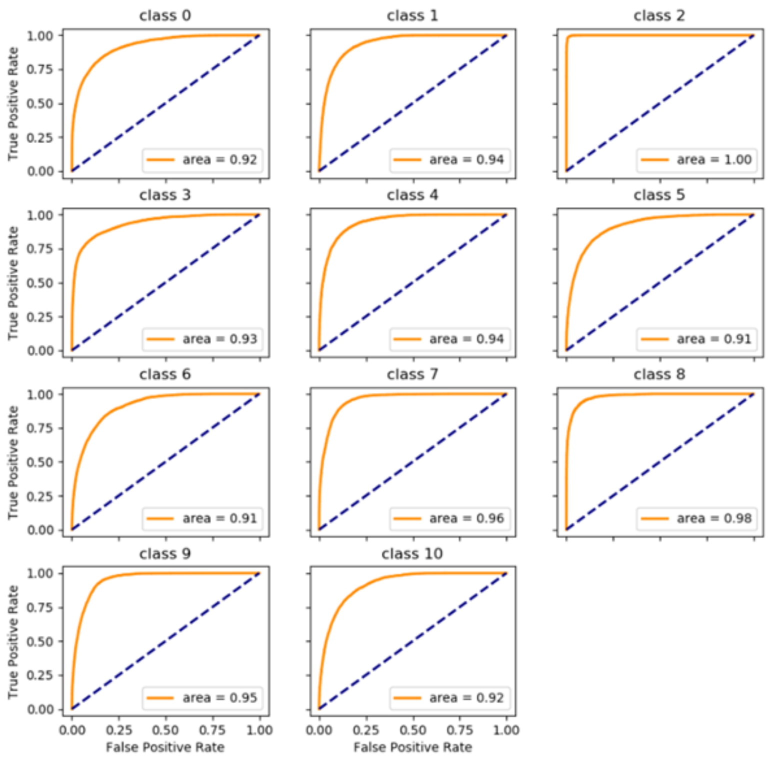


Figure 50 Strategy Predictor ROC Curves by Strategy Class

We further analyze the associated confusion matrix in Figure 51. A confusion matrix shares information about the strategy classifications that were confused with one another. That is, the confusion matrix gives detailed information about how our strategy predictor was wrong. Any entries on the off-diagonal of the confusion matrix correspond with the number errors associated with the given predicted strategy (column) and true strategy (row) occurrence. Any entries on the diagonal are correct strategy classifications. In a perfect classifier, all of the off-diagonals would be zero and the diagonal would be solid dark blue. Moreover, the darker the square, the more numbers of occurrence that the row strategy class was confused with the column strategy class.

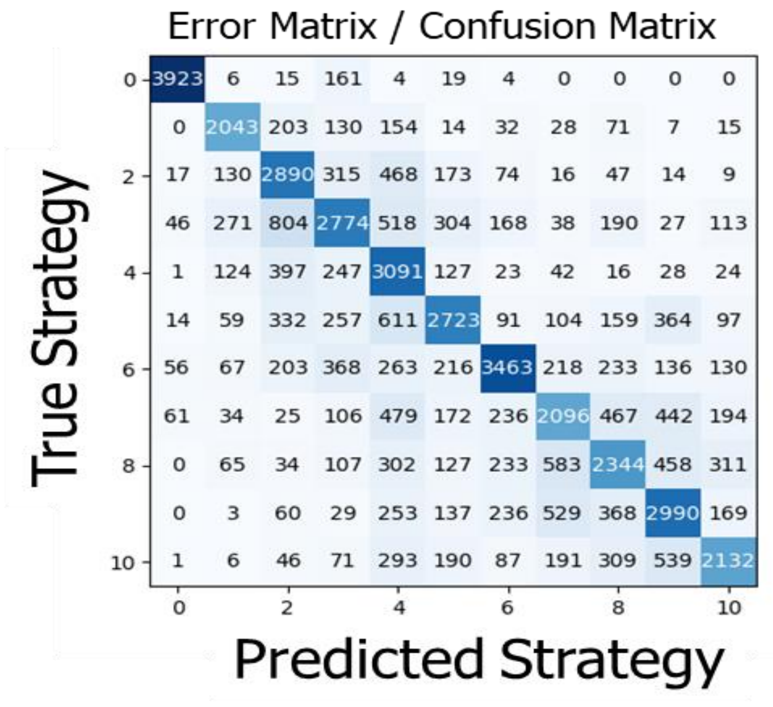
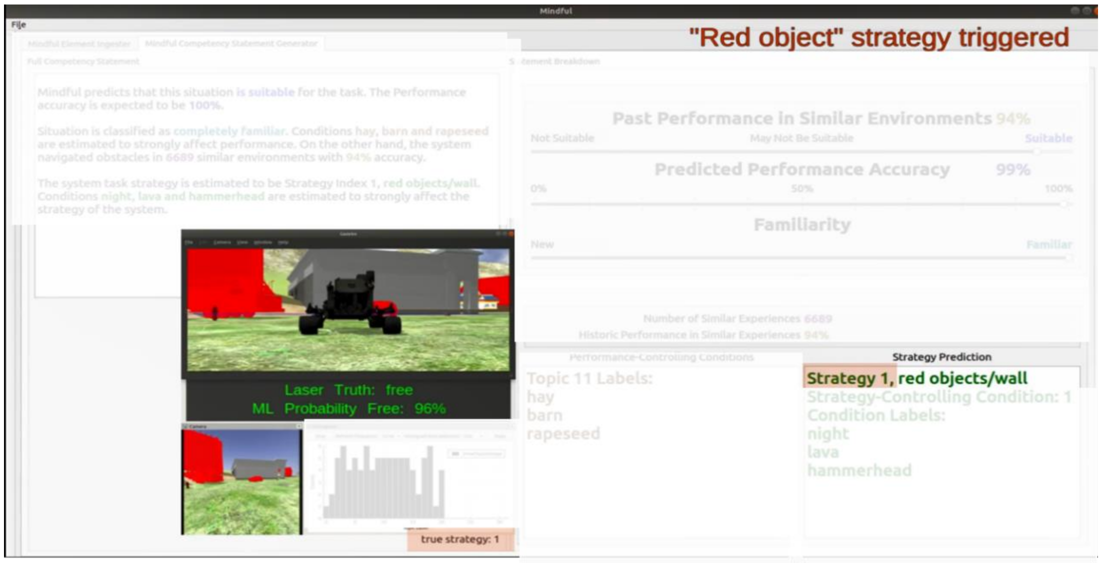


Figure 51 Strategy Predictor Confusion Matrix

A screenshot of the strategy predictor running as part of the larger MindfuL™ system is shown in Figure 52. Here, the ground robot encounters red structures and the "red objects/wall" strategy is predicted. The true strategy in the bottom left of the figure agrees with the prediction.


Mindful
File
Mindful Element Ingester
Mindful Competency Statement Generator
Full Competency Statement
Statement Breakdown
"Red object" strategy triggered
Past Performance in Similar Environments 94%
Not Suitable
May Not Be Suitable
Suitable
Predicted Performance Accuracy 99%
0%
50%
100%
Familiarity
New
Familiar
Laser Truth: free
ML Probability Free: 96%
true strategy: 1
Performance-Controlling Conditions
Strategy Prediction

Mindful predicts that this situation is suitable for the task. The Performance accuracy is expected to be 100%.

Situation is classified as completely familiar. Conditions hay, barn and rapeseed are estimated to strongly affect performance. On the other hand, the system navigated obstacles in 6689 similar environments with 94% accuracy.

The system task strategy is estimated to be Strategy Index 1, red objects/wall. Conditions night, lava and hammerhead are estimated to strongly affect the strategy of the system.

Number of Similar Experiences 6689
Historic Performance in Similar Experiences 94%

Topic 11 Labels:
hay
barn
rapeseed

Strategy 1, red objects/wall
Strategy-Controlling Condition: 1
Condition Labels:
night
lava
hammerhead

Figure 52 Online system view of an accurate strategy prediction.

While we achieved 64% on strategy classification using, considerably better than random chance (11%), where the expectation of a correct classification is equal to 1 out of 11 strategies. We note further that we could achieve ~85% accuracy if we combine confused classes cleverly (Figure 53). That is, if we combine strategy classes that are often confused with one another, we can do better at predicting online strategies. However, if we combined strategies in this way, we would lose semantic understanding of strategies at the level of granularity shown in Figure 46. In other words, these four strategy groups, which enable 85% strategy prediction accuracy, would correspond to very high-level strategies that are too high of a granularity to relate competency to the user. There would only be four strategies to provide additional insight into machine competency, reducing the level of detailed transparency into the “black box” ML system.

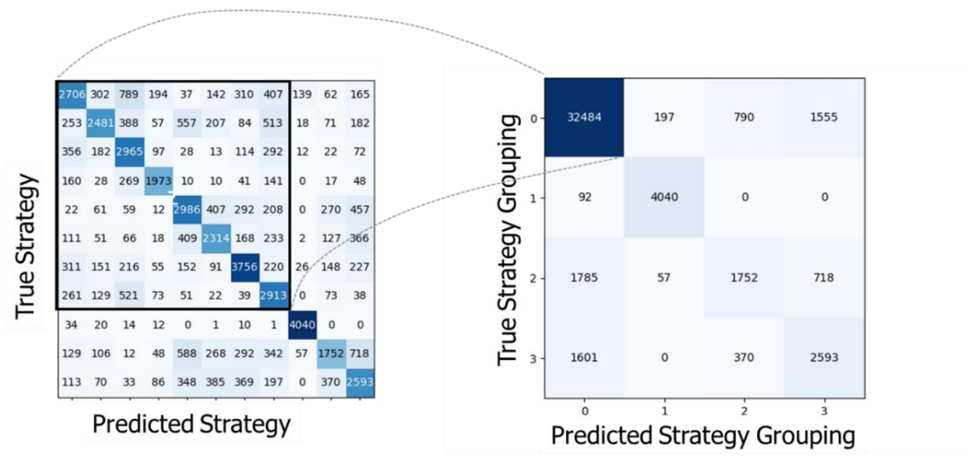


Figure 53 Clever combining of strategy classes could improve predictive performance, as we combine classes that are confused with one another

We also explored strategy prediction accuracy sensitivity to the number of clusters, neural network structures, and machine behavior definitions (experimenting with other layers shown in Figure 43). The highest performance achieved was 73% on 6 clusters, but again we sacrificed human-interpretable meaning associated with a higher granularity of strategy definition and some of this performance gain is attributable to random chance (1/6 clusters vs. 1/11 clusters).

Section 6: Online Prediction of ML Performance

Just as we layered on a predictor to go from topics to strategies, now we aim to go from topics to performance predictions, with an additional input of the ML blocked estimate. Here, we again employ a deep learning approach that trains on topic distributions and determines a binary classification of whether the machine will be correct or incorrect. In Figure 54, we show the results of our system on the training data set. Here, the green ellipse covers the case when the *MindfuL™ performance prediction* is correct (98.8%) and the red ellipse notes when it is incorrect (1.2%). Note, this is different than measuring whether the *ML system* prediction of "free" vs. "blocked" was actually correct. For these plots, label 0 represents "incorrect" and label 1 represents "correct", where we are assessing the correctness of the MindfuL™ prediction of the ML system's performance. The actual label represents whether the ML system was correct ("1") or incorrect ("0") with respect to ground truth over the incoming image while the predicted label represents our Performance Predictor output. More precisely, the ML system was correct 45,951 (summing over the "1" row) times out of 48,034 images (96%) in the training data set; it was incorrect (summing over the "0" row) 2,083 times (4%).

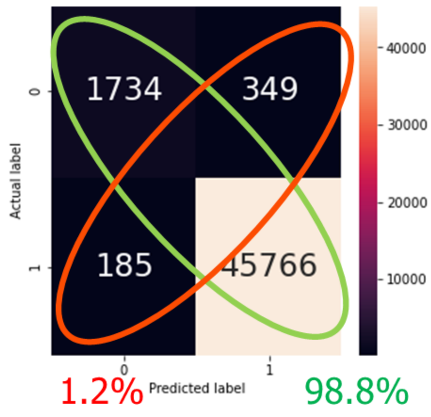


Figure 54 Performance Predictor accuracy on training data set.

The MindfuL™ system correctly predicted the machine performance over 98.8% of the training data images (green ellipse); it was incorrect 1.2% of the time (red ellipse). Here, the performance predictions worked only off of the compact topic distributions representative of each image to make its predictions, which is encouraging since we preserved enough of the information in the image through our topic distributions to assess machine competency for this task.

From the training data set to the testing data set, we note that the *ML system* itself had a considerable performance drop from 96% (on the training data) to 69% (on the testing data); these numbers are separate from the performance predictor accuracy shown in Figure 54 and Figure 55. The performance drop occurs because the testing data set was different than the training set in terms of a different array of obstacles and presence of new obstacles in the simulation environment, and likely because the ML system was over-trained in accordance with the training data set. Ongoing research is investigating the sensitivities of new environments on both ML system and Performance Predictor accuracy. However, we see that the performance predictor still produces much better results than chance (50%) on the testing data set, as shown in Figure 55. And while 68% has significant room for improvement, we are encouraged that the topic distributions are capturing enough information about the environment to make an informed (vice random) prediction on machine performance prediction.

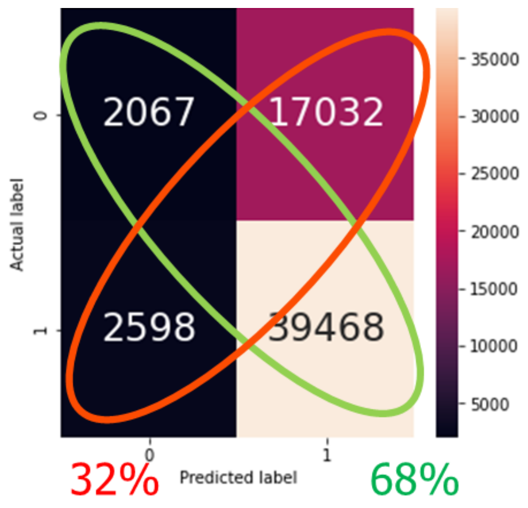
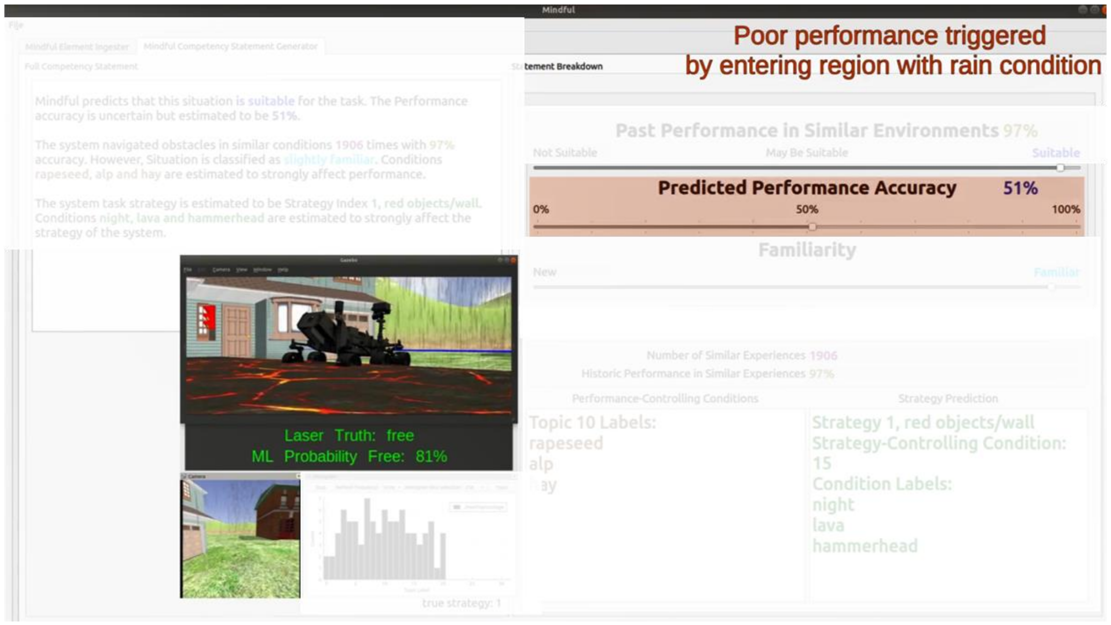


Figure 55 Performance Predictor accuracy on validation data set

The prototype Competency UI screenshot in Figure 56 shows a simulated scenario where the ground robot endures a rain environment and nears an obstacle (building) off to its left side, which are likely conditions that affect its task performance. Consistent with intuition, our performance prediction decreases in this scenario, as highlighted in Figure 56 as 51%. Both strategy and performance prediction approaches resulted in useful capabilities, considerably better than random chance. This indicates that topic distributions are capturing conditions in the environment that are important for assessing machine competency, a huge result for XAI research advancement and encouraging result for us to continue to use HDPs to compress data for later use to inform online competency prediction.


Mindful
File
Mindful Element Ingester
Mindful Competency Statement Generator
Full Competency Statement
Statement Breakdown
Poor performance triggered by entering region with rain condition
Past Performance in Similar Environments 97%
Not Suitable
May Be Suitable
Suitable
Predicted Performance Accuracy 51%
0%
50%
100%
Familiarity
New
Familiar
Laser Truth: free
ML Probability Free: 81%
Performance-Controlling Conditions
Strategy Prediction
true strategy: 1

Mindful predicts that this situation is suitable for the task. The Performance accuracy is uncertain but estimated to be 51%.

The system navigated obstacles in similar conditions 1906 times with 97% accuracy. However, Situation is classified as slightly familiar. Conditions rapeseed, alp and hay are estimated to strongly affect performance.

The system task strategy is estimated to be Strategy Index 1, red objects/wall. Conditions night, lava and hammerhead are estimated to strongly affect the strategy of the system.

Number of Similar Experiences 1906
Historic Performance in Similar Experiences 97%

Topic 10 Labels:
rapeseed
alp
hay

Strategy 1, red objects/wall
Strategy-Controlling Condition:
15
Condition Labels:
night
lava
hammerhead

Figure 56 Online view of intuitive performance prediction

Section 7: Online Prediction of Strategy- & Performance-Controlling Conditions

In order to explain the conditions (why) associated with our predictions of machine strategy and performance predictions, we employ feature importance methods to find which HDP topics contribute the most to each competency prediction. Ideally, the important topics we find can be used to provide semantic understanding to the user based on why the machine arrived at its result. For our obstacle recognition task, a topic may represent that an obstacle is present like "building" or "wall". For our airplane recognition use case, a topic might correspond with an "airport", "runway" or the presence of "wings" or a "nose". For now, we can still leverage the presence of certain topics regardless of whether they carry semantic meaning, such as "Topic 1" or "Topic 2" conditions by comparing across different tasks. We refer to topic expression as "high" or "low" when the topic weight is more than two standard deviations higher or lower than the mean, and we are considering alternatives to this threshold.

In some cases, human connotation might be matched with high or low topic expressions through manual inspection. In others, there may not be any human-discernable correlations. Even if there is no human-understandable context, we can group conditions, strategies, and performance together based on common results. For example, we can construct an abstract rule such as "If Topic 3 is highly expressed, then the machine employs Strategy 4 with predicted performance of 95%." Then, even though the human might not understand what Topic 3 or Strategy 4 are, he/she can compare similar occurrences of this rule to one another to identify common themes or gain some insights into the "black box" behavior, if only at an abstract level. Again, we ideally aim to attach semantic meaning to both conditions and strategies. A more useful-to-humans rule abstraction for competency would be “If Topic 3: ‘airport’ is highly expressed, then the machine employs Strategy 4: ‘looked for wings’ with predicted performance of 95%”.

To find these conditions which influence the strategy and machine performance predictions, we employ Shapley Additive exPlanations (SHAP) [62]; in our case, SHAP correlates specific topic expressions to the strength that they contribute to the predicted performance (see below for a mathematical description). More generally, Shapley feature values are computed leveraging coalitional game theory, where for a given instance of data, the feature values (or group of features) are treated as a single player. The contribution that this player has in shifting the prediction from the average value, is the Shapley feature value. We apply SHAP to the HDP topic distribution for a given image and use the magnitude and direction to determine which topic values improved and worsened performance by what quantity from average for a given data instance.

Shapley values [19] have traditionally been used to determine how to divide the payoff or gain amongst a group of players in a cooperative game. It measures the average marginal contribution from every sequence of players entering the game, evaluated with and without the j^{th} player. In our case, the players are the topics themselves, derived from HDP processing of images described in Section 2, such as Topic 1: “wings” or Topic 3: “airport”. The game is either the strategy prediction (Section 5) or the performance prediction task (Section 6), depending on whether we are estimating strategy-controlling or performance-controlling conditions. The “gain” is the prediction for a particular image instance minus the average prediction for all instances. And the Shapley Value is the average change in the competency prediction when another topic value is added, given a competency prediction based on an existing group of topic values, as described in the previous two sections.

We are interested in how each topic affects the competency prediction of an image. In a linear model it is easy to calculate the individual effects. For example, here is what a linear model prediction looks like for one data instance [62] (Eq. 23):

$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \quad (Eq. 23)$$

where x is the instance for which we want to compute the contributions. Each x_j is a topic value, with $j = 1, \ldots, p$. The β_j is the weight corresponding to topic j.

The contribution φ_j of the j-th topic on the prediction $\hat{f}(x)$ is (Eq. 24):

$$\varphi_j(\hat{f}) = \beta_j x_j - \beta_p E(X_j) \quad (Eq. 24)$$

The contribution is the difference between the topic effect minus the average effect. Now we know how much each topic contributed to the prediction. And if we sum all the topic contributions for one instance, the result is the following (Eq. 25):

$$\sum_{j=1}^{p} \varphi_j(\hat{f}) = \sum_{j=1}^{p} \left(\beta_j x_j - \beta_p E(X_j)\right) = \hat{f}(x) - E\left(\hat{f}(X)\right) \quad (Eq. 25)$$

However, this is only applicable to linear models. Shapley values provide a way for us to do feature importance analysis over general models [62], such as the neural network and random forest models used by our Strategy and Performance Predictor, respectively. Shapley value is defined as (Eq. 26):

$$\varphi_j(val) = \sum_{S \in \{x_1, \ldots x_p\} \setminus \{x_j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(val\left(S \cup \{x_j\}\right) - val(S)\right) \quad (Eq. 26)$$

Where *val* is the prediction for feature values in set *S* that are marginalized over features that are not included in *S*, p is the total number of players, and x_j represents the value (topic weight) associated with player (topic) j. In our case, this means computing the competency predictions without some of the topics. To compute Shapley values, we simulate that only some topics are playing ("present") and some are not ("absent").

The SHAP method leverages Shapley values to specify an explanation (Eq. 27):

$$g(z') = \varphi_0 + \sum_{j=1}^{M} \varphi_j(z'_j) \quad (Eq. 27)$$

where g is the explanation model, $z' \in \{0,1\}^M$ is the coalition vector, *M* is the maximum coalition size and $\varphi_j \in \mathbb{R}$ is the feature attribution for a feature *j*, the Shapley values [62].

As an alternative to SHAP, Gini Importance or Mean Decrease in Impurity calculates each feature importance as the sum over the number of splits (across all tress) that include the feature, proportionally to the number of samples it splits. This is a common ante-hoc XAI method used to determine general feature importance for random forest tree models.

Unlike Gini importances, that are useful for determining the general importance of features, SHAP supports per-instance feature importance, which is necessary for our near real-time competency assessment framework. For example, in our airplane recognition task, it is more useful to understand why the ML system thinks there is an airplane in the current image that it is processing, such as “airport” topic being highly expressed than a generality that an “airport” is an important feature in general for identifying airplanes in an image. SHAP gives the former on a per-image basis, as desired. Gini importances give the latter in a general sense over a set of images. Figure 57 shows both the Gini importances and SHAP values for our performance predictor. Each point represents a Shapley value for a topic and performance prediction. The color represents the actual topic weight from low (blue) to high (red). For example, if Topic 3 corresponds to “airport”, high expression of Topic 3 would indicate that an airport is present in the image; low expression of Topic 3 would indicate that no airport is present in the image.

Gini importances generated from the random forest model agree with top two SHAP channels, supporting the SHAP approach validity. Other than the ML blocked estimate, there is little variance in both SHAP values and Gini importances, and we suspect this is likely due to needing all topics in order to make an informed competency prediction. Future work is attempting to gain separation between topics across features that are important for performing the ML task itself, rather than a compressed distribution over all of the features in the image. In our airplane recognition task, this would mean that we want a topic to separate specifically over the presence of “airport”, and not contain overlapping features associated with that topic, such as “grass” or “roads” that are not as well-associated with the ML task. Automatic and reliable extraction of competency-relevant features from data at a summarized level of granularity and semantic meaning is an open research problem, currently being addressed by the author. In Figure 57, a large blocked estimate negatively affects incorrect classification, noted by the yellow star, thereby yielding positive effect on correct classification. Moreover, when Topic 7 is not expressed, it could positively or negatively impact classification, as indicated by the orange stars; these could be rare events corresponding to images with unique properties.

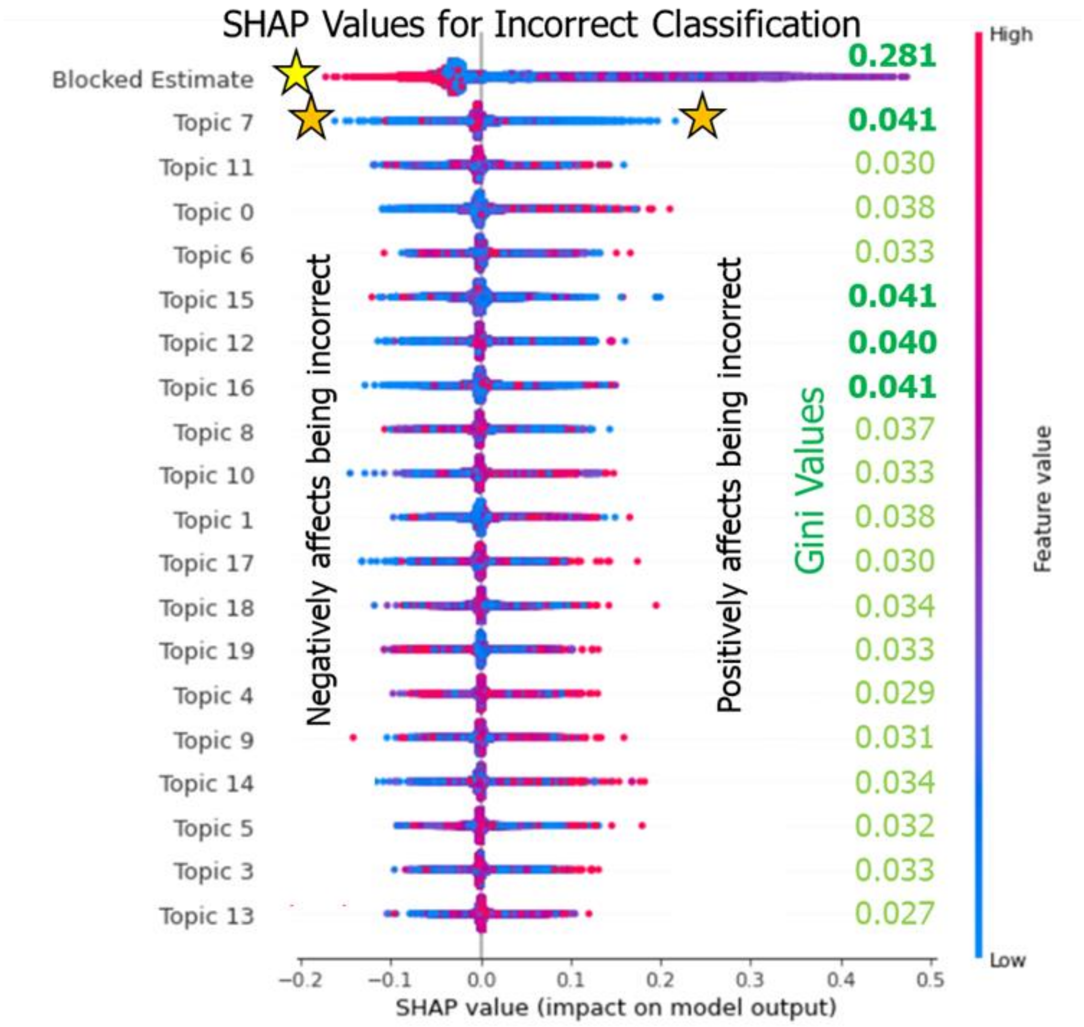


Figure 57 SHAP plot for incorrect performance prediction

And when we filter on high expressions of Topic 7, we visually can see indicators of these rare edge cases, as shown in Figure 58.

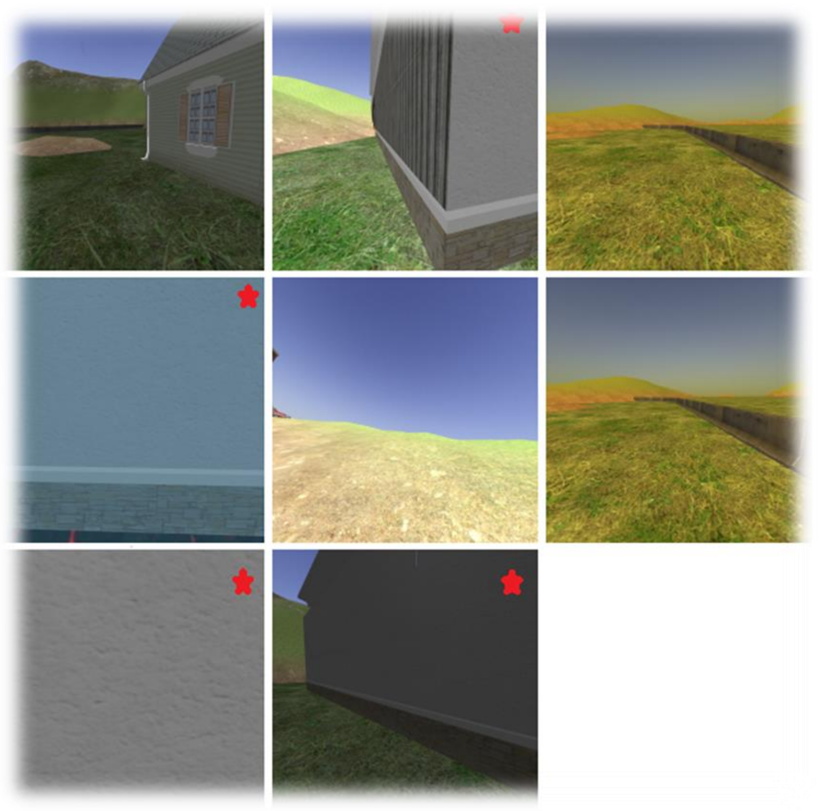


Figure 58 High expressions of Topic 7 corresponds to unusual lighting and rare obstacle proximity events (staring at wall with homogeneous features)

To compute *strategy*-controlling conditions, different from the *performance*-controlling conditions found above, SHAP values are generated for each strategy classification. The SHAP plot for Strategy 5 is provided in Figure 59. Gini importances generated from a random forest alternative model are again consistent with the top SHAP topics. Moreover, the top 5 important topics are preserved across all strategy classifications. This means that five topics encode the most distinguishable information over all strategy predictions. Notably, high expressions of Topic 16, 19, and 0 positively contribute to an image clustered into #5 (black stars), while Topic 15 and 1 negatively contribute (orange stars). For example, in our airplane recognition task, we would expect that the high expression of Topic 1: “wings” would positively contribute to Strategy 3: “looked for airplane wings”.

Next, we attempt to discern semantic meaning from manually inspecting images associated with high expressions of Topics 16, 19, 0, 15, and 1 to see if we can determine logical reasons why they might be affecting the machine's propensity to employ Strategy 5: "shadow detection".

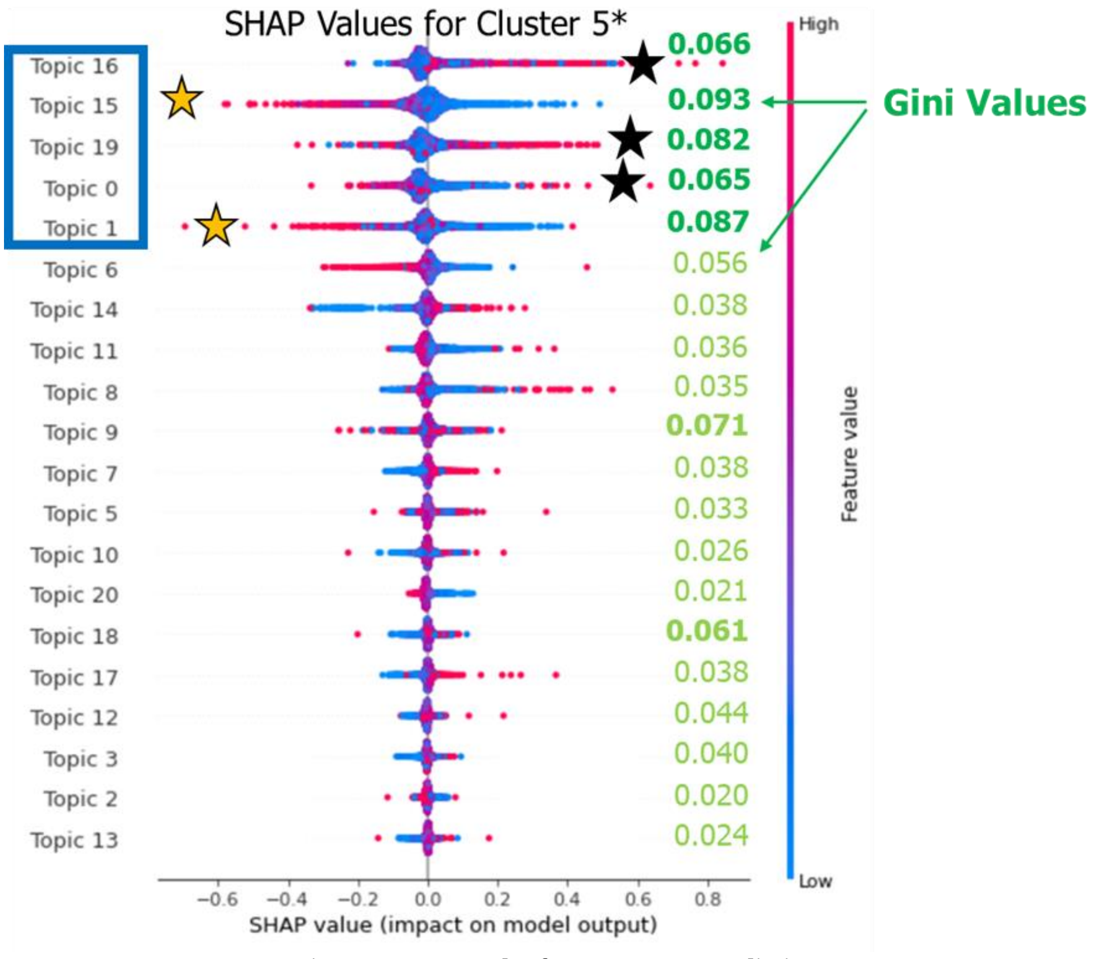


Figure 59 SHAP plot for Strategy 5 prediction

And when we look at the data associated with high expressions of these topics, we make some useful observations. Topics 1 and 15 correspond with dark images, making intuitive sense for distinguishing the Strategy 5: "shadow detection" strategy from other strategies. Topic 19 seems to correspond to buildings, where the building acts as a close obstacle and Topic 0 seems to consistently show a jersey barrier wall, which was consistent with one of our human-interpretable strategies (Strategy 8). However, Topic 16 doesn't have any discernable human context when parsing through images that have highly variable environmental features present in the filtered data set. Here, we point out importantly that there are no guarantees that the machine cares about what humans can distinguish by their eyes. In other words, there might be commonalities in these data that the machine recognizes as important for determining strategy that humans do not think would be intuitively important. A powerful aspect of our approach is that we can still recognize these conditions, even though they may only be identifiable by a common label (i.e. Topic 16), and not interpretable further semantically.

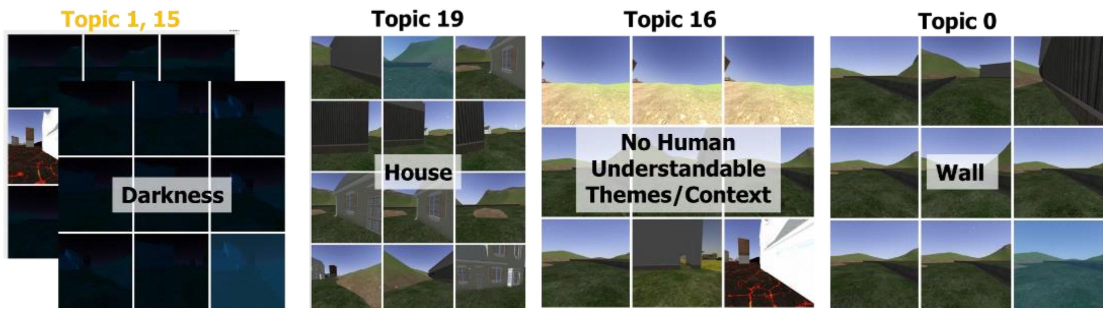


Figure 60 High expressions of Topic 16, 19, and 0 positively contribute to correct clustering

In summary, our online competency-aware system uses a deep explainer method, SHAP, to determine which conditions were most influential to the current strategy and performance predictions for the current environment.

Section 8: System Training and Online Framework

8.1 Walkthrough of Offline Training System Diagram

Offline, we train our competency predictor components through a series of training regimes. At a high level, this is how the system works to train all of its components. The training is sequential in nature, but is also automated so we can train on a new dataset with minimal human involvement. The MindfuL™ offline system is illustrated in Figure 61 and the steps associated with the diagram are described below.

1) We train the experience encoder to ingest data and produce topic models; during this phase, the HDP determines the features associated with each topic and the number of topics that sufficiently explain the training data set.
2) Topic distributions associated with each document in the training data set are stored in the Memory Bank.
3) Labels ingested by the Element Ingester are linked to data in the Memory Bank.
4) We determine ground truth from comparing ML estimates over the sensor data to the LiDAR data obtained from the simulation.
5) We train a performance predictor to go from topic models to “correct” or “incorrect” predictions, giving rise to our trained Performance Predictor model.
6) We define strategies using the Behavior Clusterer method used in both Chapters 3 & 4 based on the underlying behaviors gleaned from the ML system activations.
7) We train a Strategy Predictor to learn relationships between topic models and strategies, giving rise to a trained Strategy Predictor model.

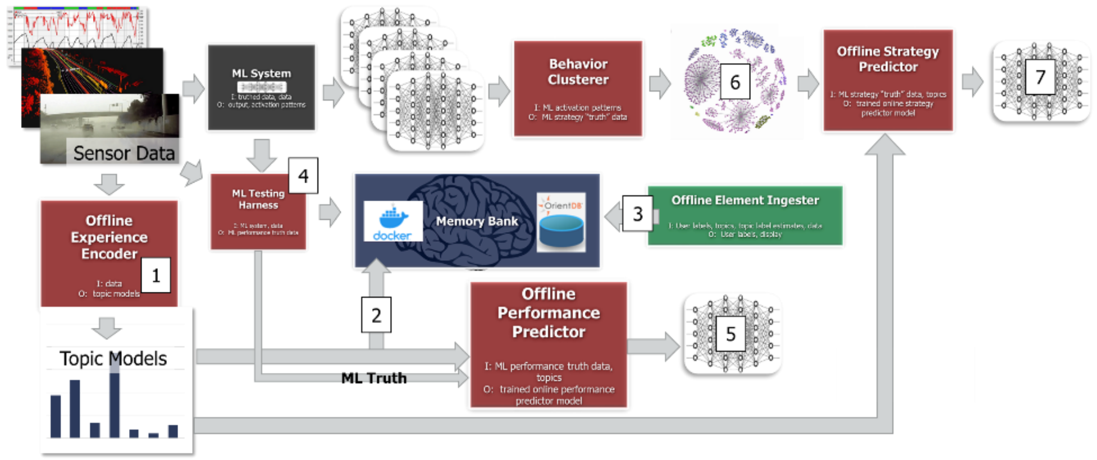

Sensor Data
ML System
I: truthed data, data
O: output, activation patterns
Behavior Clusterer
I: ML activation patterns
O: ML strategy "truth" data
6
Offline Strategy Predictor
I: ML strategy "truth" data, topics
O: trained online strategy predictor model
7
ML Testing Harness
4
docker
Memory Bank
OrientDB
3
Offline Element Ingester
I: User labels, topics, topic label estimates, data
O: User labels, display
Offline Experience Encoder
I: data
O: topic models
1
2
Offline Performance Predictor
I: ML performance truth data, topics
O: trained online performance predictor model
5
Topic Models
ML Truth

Figure 61 Offline system training phase diagram

8.2 Walkthrough of Online Competency-Aware System Diagram

Both our offline and online functionally distributed architectures supported analyses of alternative approaches easily due to a disciplined, function-based input/output implementation. This was especially important when considering alternatives to our HDP, SHAP, and Deep Neural Network approaches to predictor components. Details of auto-encoder, feature importance, and random forest models are omitted from this dissertation, but were explored by the MindfuL™ team during system development. Online, we provide competency information to the user in accordance with involved component interactions. These interactions are summarized in Figure 62 and described in the steps outlined below.

1) Data is ingested and the Experience Encoder infers topic distributions based on the learned HDP model, wherein features associated with each topic and the number of topics is constant.
2) Inference is performed over the trained Strategy Predictor to go from topics to a Strategy estimate.
3) Inference is performed over the trained Performance Predictor to go from topics to a Performance estimate.
4) SHAP methods determine the most important conditions (topics) for both strategy and performance predictions.
5) The number of environments and the historical competency is determined by the Environment Similarity Calculator component by comparing the current topic model to those compactly stored in the Memory Bank.
6) The Ingester & Interpreter components support user-provided labels.
7) The Competency Statement Generator produces long-form statements.
8) The Information Analyzer makes a determination of whether to allow the machine to perform the task or whether user intervention is recommended.
9) The Competency UI displays all of the competency information derived above to the user.

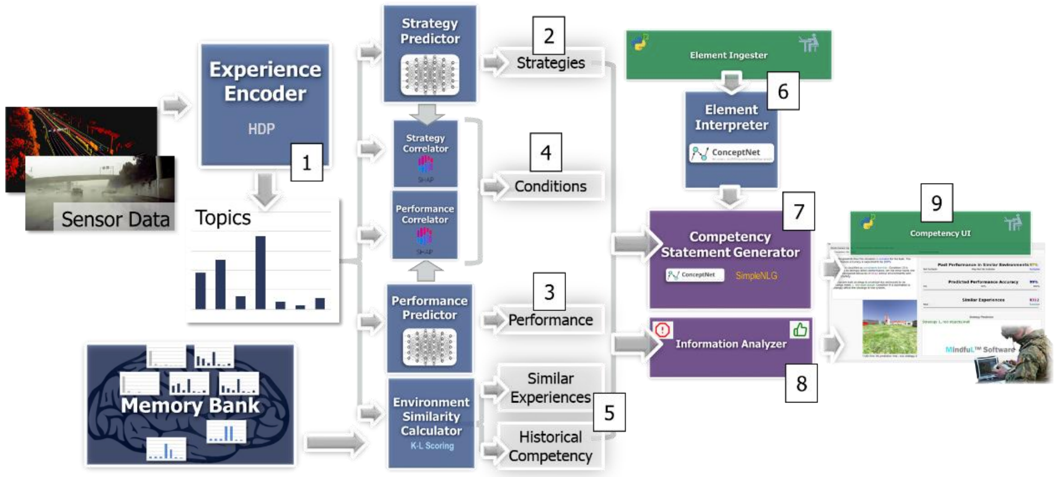


Figure 62 Online system competency assessment diagram

8.3 Near real-time competency assessment offers "online" utility to users

The online system supports near-real time operation, and the processing times for several of the components is provided in Table 13. MindfuL™ competency assessments update every two seconds, which is a cadence faster than human preference for current applications. The competency statement generator remains the most expensive functionality, typically accounting for more than half of execution time per frame. This is largely because the environment similarity algorithm searches through all past experiences in the agent's memory within a function call in this component. Fortunately, this computation can be parallelized to realize even faster update rates. Moreover, we could intelligently maintain a memory bank where we discard experiences that are sufficiently similar to one another to speed up the calculations.

Table 13 Computational times associated with several components for two sample updates

Component	**Update Rate with 1k Image Set**	**Update Rate with 10k Image set**
Experience Encoder (HDP)	0.204	0.738
Performance Predictor	0.040	0.037
Performance Correlator	0.001	0.0004
Environment Similarity Calculator	0.234	0.140
Competency Statement Generator	1.076	0.919
Memory Bank	0.010	0.348
System	**1.556**	**1.83**

Section 9: Interpretations

In Chapter 4, we showed the utility of a competency-aware machine learning approach to provide near real-time competency information to a user. We capitalized on benefits hypothesized in Chapters 2 & 3 by realizing an online predictive capability for strategy and performance (competency). We implemented an automated way to devise conditions, strategies, and performance with no human involvement required other than to improve semantic interpretability. To determine conditions, we leveraged shared descriptor characterization from the HDP approach and SHAP feature importance analysis to determine which conditions (HDP-derived topics) were most influential for our online competency predictions. Our online competency predictions performed well above chance with 64% accuracy on strategy predictions (compared to 11% chance) and 69% performance predictions (compared to 50% chance) when using a generalizable HDP approach to compactly describe inputs. Importantly, this result showed that HDPs captured competency-relevant information, in spite of their incredible compression benefits (1 year of video data is compressed to just 2TB of data). Finally, we showed timeliness of our approach to scale to near real-time as online competency predictions took less than 2 seconds to compute over 10,000 previous experiences, faster than a user would desire for an obstacle recognition task. In conclusion, these results and those in Chapter 3 attest to the generalizability and applicability of this approach to other AI systems and applications. Research in this chapter contributes to increased AI interpretability and competency-awareness needed to increase trust and transparency of AI systems.

Chapter 5: Conclusions and Future Work

Section 0: Conclusions

In conclusion, this research provides materials and methods to study and assess the competency, including how (strategies), why (conditions), and the expected result (performance), of otherwise "black box" machine learning systems. First, we manually identified important relationships between environmental conditions and emerging strategies on deep learning performance in a 2-d pursuit game application in Chapter 2, driving further exploitation in Chapter 3 and motivating predictive approaches in Chapter 4. Moreover, we demonstrated the effectiveness of an agent trained with the DDPG reinforcement-learning algorithm in a speed-overmatched pursuit game with uncertain target information. The resulting RL agent outperformed a baseline bearing-following strategy by increasing capture successes by more than 100% in a 5000-trial experiment and was more robust to harsher pursuit game conditions. No prior research approach addressed speed overmatch, uncertainty, and dynamic speed control with deep learning or any other control system for a 2-d pursuit game. Finally, we discussed the potential utility for leveraging historical competency for online unmanned system control. We manually observed separation of learned pursuit behaviors into strategy groups. And we manually hypothesized environmental conditions that affected performance. These manual hypotheses regarding machine competency motivated automated abstraction of conditions, performance and strategy relationships investigated in Chapters 3 & 4.

In both Chapters 3 & 4, we found that neural network activation patterns could be abstracted into human-interpretable strategies for two separate deep learning approaches, including the pursuit game application analyzed in Chapter 2. In Chapter 3, we found that compact representations of activation patterns (behaviors) preserved more information relevant to machine performance than the machine's actions themselves. Moreover, we found that clusters of behaviors (strategies) separated naturally over pursuit trajectories that were relatable to humans and that strategies could be exploited as predictors for machine performance. And finally, we defined machine *commitment*, and found that it was significantly higher in games with Win outcomes than Loss outcomes. These results motivated further exploration of online competency prediction approaches.

And in Chapter 4, we realized online prediction capabilities for condition, strategy, and performance competency assessments. The HDP approach encoded information pertinent to the machine learning task and system, as evident by the success of strategy and performance predictor components. Furthermore, the HDP encoded data compactly, supporting scalability of the approach to handle a large library of machine experience in our memory bank database. Our approach to automated condition determination using SHAP, while difficult to assess in terms of ground truth, yielded useful results consistent with human intuition. Our prototype strategy and performance prediction components produced useful competency estimates, significantly outperforming random chance. Moreover, our environment similarity calculator adequately determines whether a machine is familiar with the current task, and quantifies the number of similar experiences the machine has been trained on in the past; and lastly, as a byproduct of its approach, it shows promising results for detecting faulty sensor and other anomalous input data.

Since all of the prediction components are trained based on a common topic representation (and not the input data directly), only the HDP component needs to be modified to support competency prediction over other types of data. Moreover, only the offline strategy definition needs to be modified to adapt to new ML systems. Since only two of the 14 components require significant modification for new applications and systems, the resulting MindfuL™ system is highly portable. Therefore, the approach can be applied to a large number of machine tasks and systems. By uncovering relationships between environmental conditions, machine strategies, & strategies and by giving rise to online estimation of machine competency, we increase transparency and trust in machine learning systems, contributing to the overarching XAI initiative.

Section 1: Broader Implications & Future Work

For the 2-d pursuit game application, the RL approach should be tested in a higher fidelity simulation and integrated with an unmanned system for testing in the real world. Additionally, it can be extended to three dimensions for further applicability to missile & space domains, for example.

Regarding XAI, MindfuL™ software provides insights into how, why, and the expected result of an ML system for a particular task. This has many benefits regarding trust and transparency, and broader implications for the future of manned and unmanned teaming. Trust by humans is gained into the ML system under assessment because the MindfuL™ system identifies when an ML system is likely to succeed or fail. Transparency is gained by understanding why and how the ML system performs its task a certain way. For example, in an “L-shaped” maneuver in the pursuit game, the human could doubt that the ML system was on track for a capture, even though the performance prediction was high. But if the human is aware of the internal strategy of the ML system being consistent with and “L-shaped” maneuver, then he/she can increase trust in a likely successful capture. Moreover, if the human is teamed with the machine and knows about the “L-shaped” maneuver, he/she can choose a complementary action or maneuver with strong confidence and anticipation that the machine is going to make a 90-degree turn within some timeframe.

In order to anticipate the maneuver, the machine must be thinking ahead and have a plan for the L-shaped strategy prior to making its 90-degree turn. Future work could examine activation patterns and attempt to understand the time in which an agent commits to a certain strategy or is planning to execute a certain strategy based on certain neuron activation values. The raw activation plot in Figure 63 for six Strategy 9: “L-shaped maneuver” games is given here as further motivation for future research. There are not obvious correlations between the activation patterns beyond higher numbers of active neurons than other sample games examined by the eye. Example military use cases for complementary human-machine tactics include small unit maneuvers with UxV teammates or fighter jet dogfighting with UAV teammates.

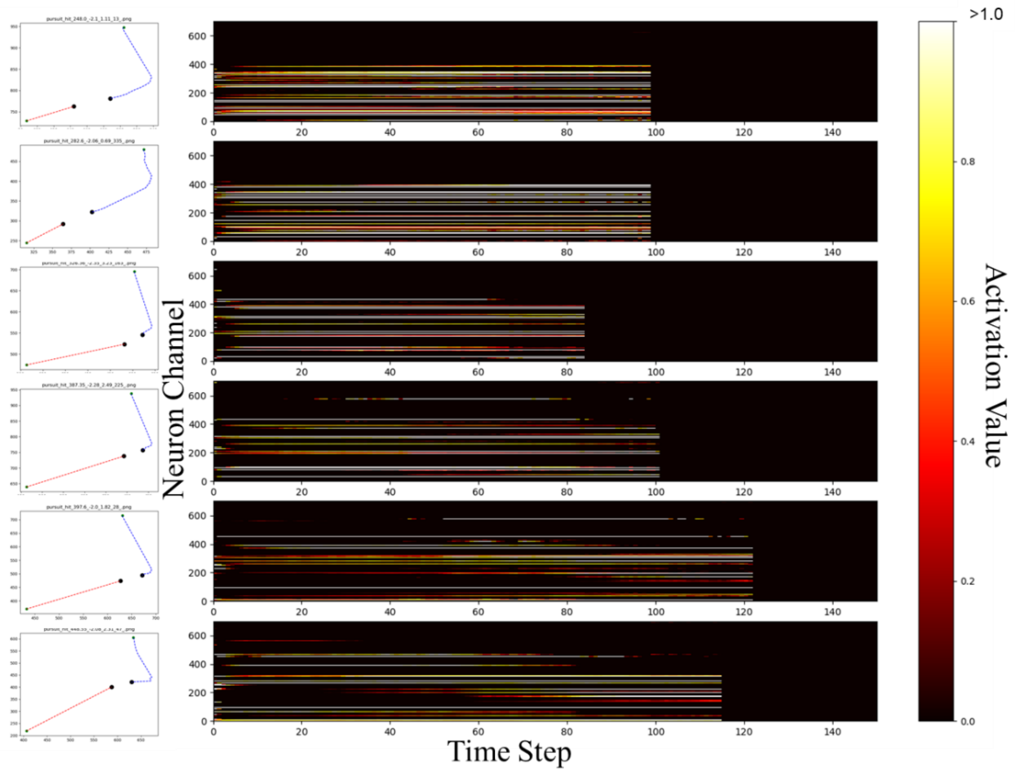


Figure 63 Raw activation patterns for L-shaped maneuver pursuit games

Separately, competency awareness also helps with energy preservation for deployed unmanned systems without human teammates or controllers. Performance prediction could help a pursuer agent, for example, determine whether it should pursue the target of opportunity, wait for a different target, or avoid resource expenditure under unlikely success situations; examples include expending energy to attempt to intercept a pass in sports or expending limited ballistic missile defense resources toward incoming threats. Such analysis would also help determine the timing for when the pursuer should begin pursuit, leading to better energy efficiencies for unmanned systems with endurance limitations.

Future work should consider supporting a multi-modal HDP that ingests unorthodox sensory inputs like labels and machine behaviors (activation patterns) themselves in order to increase competency prediction efficacy. Such inputs are different than what many consider orthodox sensory inputs, like those from cameras or radar sensors. The diagram in Figure 64 illustrates the pipeline for training an HDP on multiple sensory modalities. In this case, features are extracted from each data stream and summarized via feature counts. Then, the topic model looks at the distribution of feature counts in the data and determines the topic distribution associated with that input data. One important property of HDPs is that they can handle missing data. In this case, if one of the sensor data streams is missing, I can still estimate a topic distribution and compare that to other samples where the data stream was present.

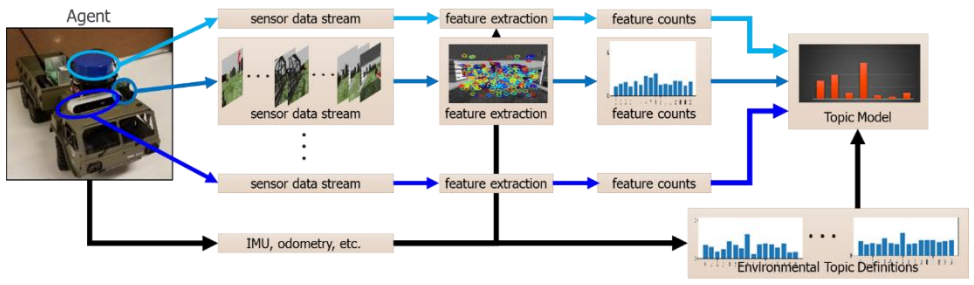


Figure 64 Processing of Multi-Modal Sensory Inputs through HDPs

Additionally, an online mechanism to estimate machine commitment *in situ* could be exploited for machine predictability and support human-machine teaming. Moreover, future research, especially for human-machine teaming applications, should focus on the semantic interpretability of machine-derived conditions and strategies, so that a human teammate can understand the results more easily. And, in cases of no human partners, a closed-loop controller should supervise ML behavior and intervene with stoppages or fail-safe policies when unacceptable performance estimates are estimated. We also note that while the UMAP representation of strategies is good when it is trained on all of the data, it suffers to fit new data into existing manifolds properly. This became a concern during use of the online predictive capability when ingested images appeared to humans to correspond to a different strategy than what was assigned as the true strategy. Further investigation into ground truth definition is underway to resolve this potential issue. However, we only expect to see improved Strategy Predictor performance and note that the current system provides results that make sense to the user.

The Information Analyzer output currently supports just a binary interpretation: “Continue normal operation” vs “manual override recommended”, as shown in Figure 65. In the future, MindfuL™ software could produce suggestions for other actions and provide other alerts regarding system competency. Such actions could span sensor modality selection or sensor acquisition of new data, beneficial for refining competency assessment or suggestions to modify a selected action, like reducing speed in a ground autonomy application, to maintain acceptable mission performance. We could also prompt labeling actions for attaching semantic meaning to an unlabeled Condition. These additional suggestions necessitate innate awareness of performance and strategy predictions, their reasons (conditions) and their uncertainties, and should be paired with a Value of Information (VoI) approach for more robust results.

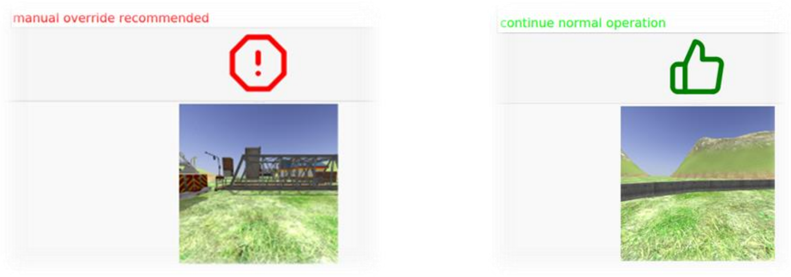


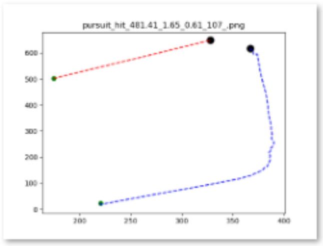
Figure 65 Suggestions for user intervention

Finally, as the MindfuL™ approach was careful to be input- and ML system-agnostic, the XAI capabilities should be exercised for other machine tasks and input data types. User experience should be evaluated and interfaces updated accordingly.

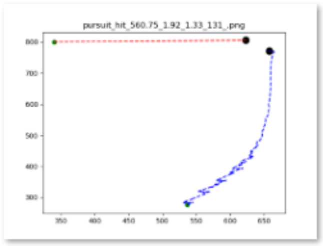
Appendix

<u>Section 0: Sample Pursuit Games from Automatically-Derived Strategies</u>

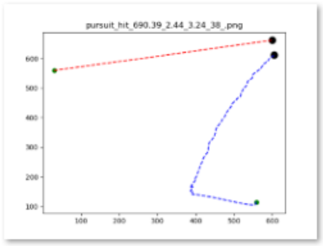
Nine sample games are provided at random from each strategy grouping in the figures that follow. The metadata in the filename corresponds to the game outcome, initial distance to the target (meters), initial angle to the target (radians), maximum target speed (meters per second), and game episode identifier parameters, delineated by an underscore, respectively.



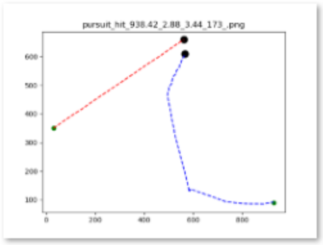
pursuit_hit_481.41_1.65_0.61_107_.png



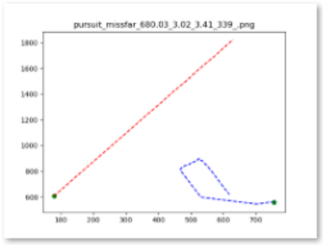
pursuit_hit_560.75_1.92_1.33_131_.png



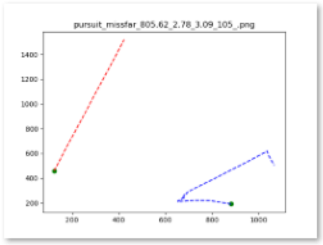
pursuit_hit_690.39_2.44_3.24_38_.png



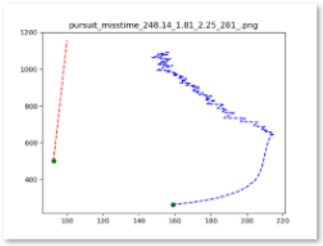
pursuit_hit_938.42_2.88_3.44_173_.png



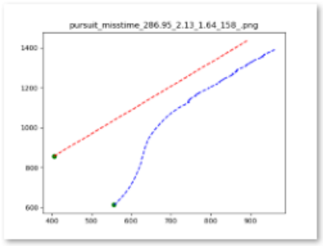
pursuit_missfar_680.03_3.02_3.41_339_.png



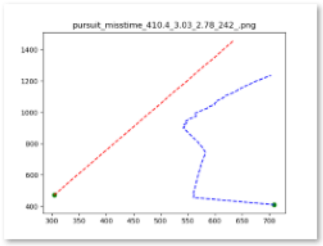
pursuit_missfar_805.62_2.78_3.09_105_.png



pursuit_misstime_248.14_1.81_2.25_281_.png



pursuit_misstime_286.95_2.13_1.64_158_.png



pursuit_misstime_410.4_3.03_2.78_242_.png

Figure 66 Strategy 0 Game Samples

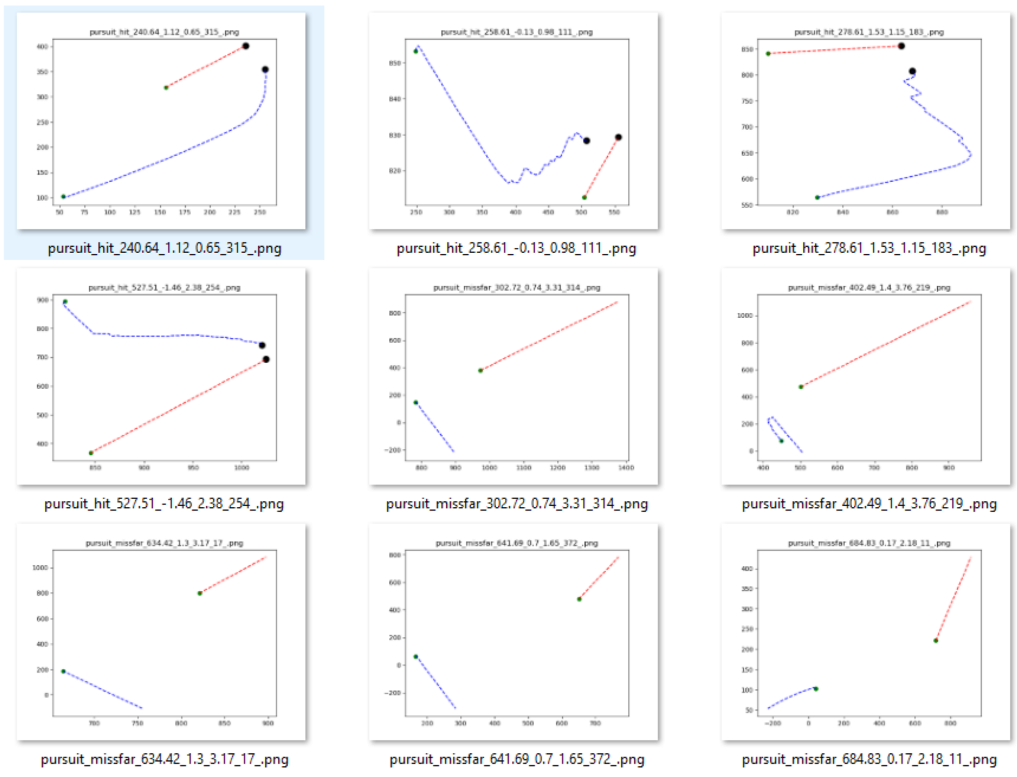


Figure 67 Strategy 1 Game Samples

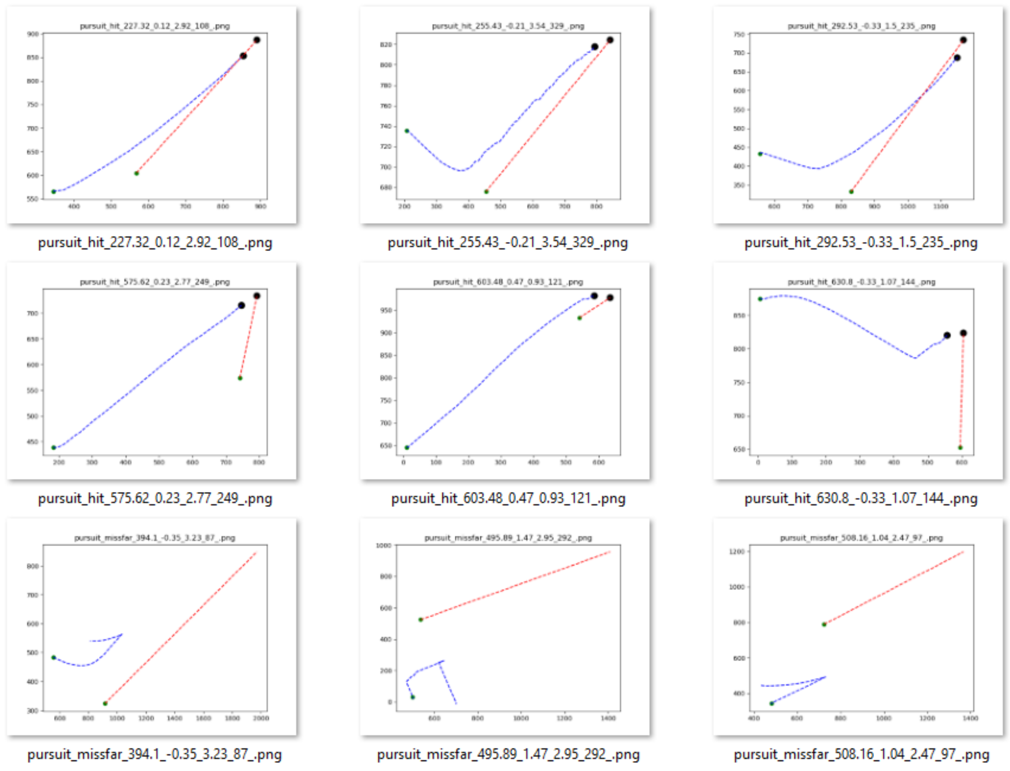


Figure 68 Strategy 2 Game Samples

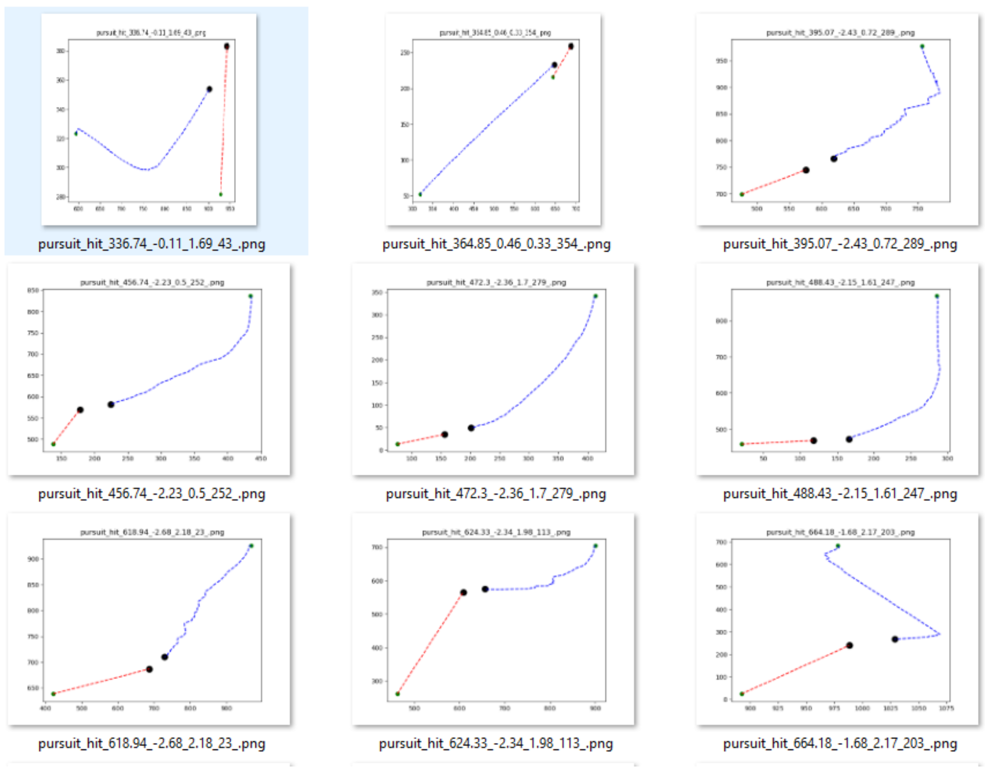


Figure 69 Strategy 3 Game Samples

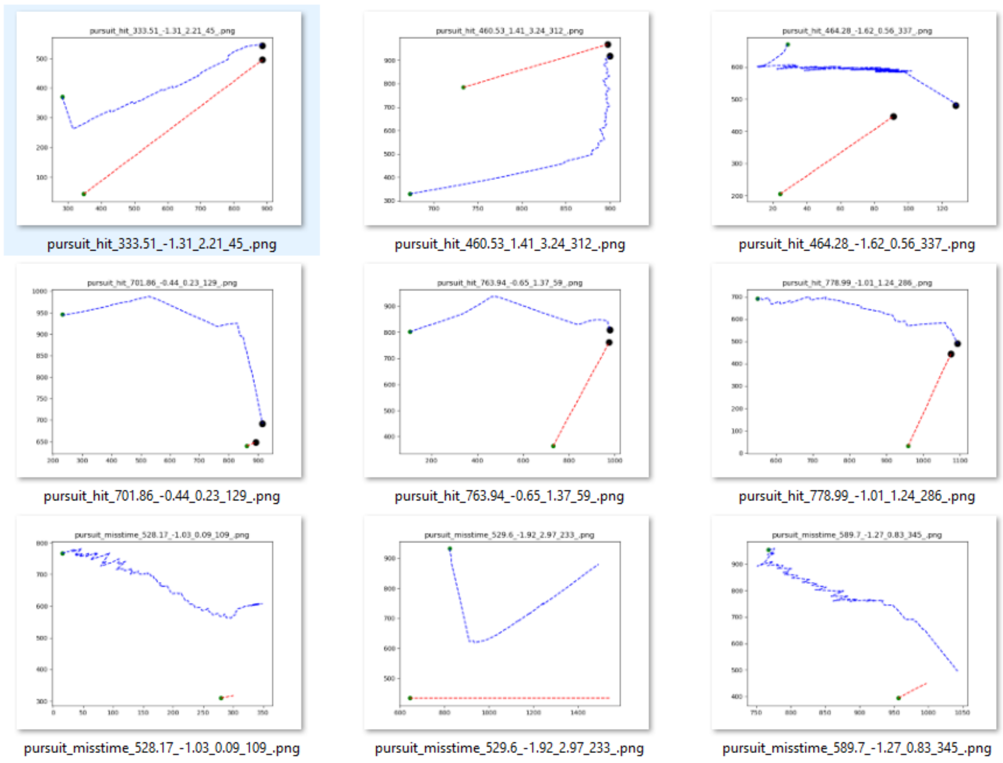


Figure 70 Strategy 4 Game Samples

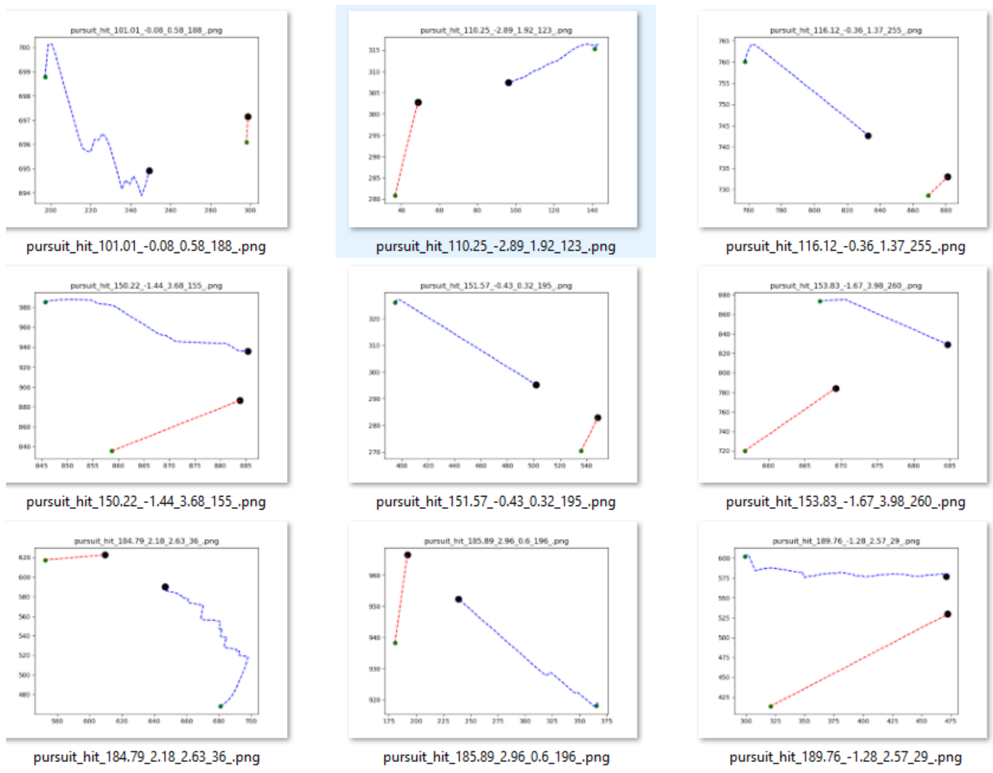


Figure 71 Strategy 5 Game Samples

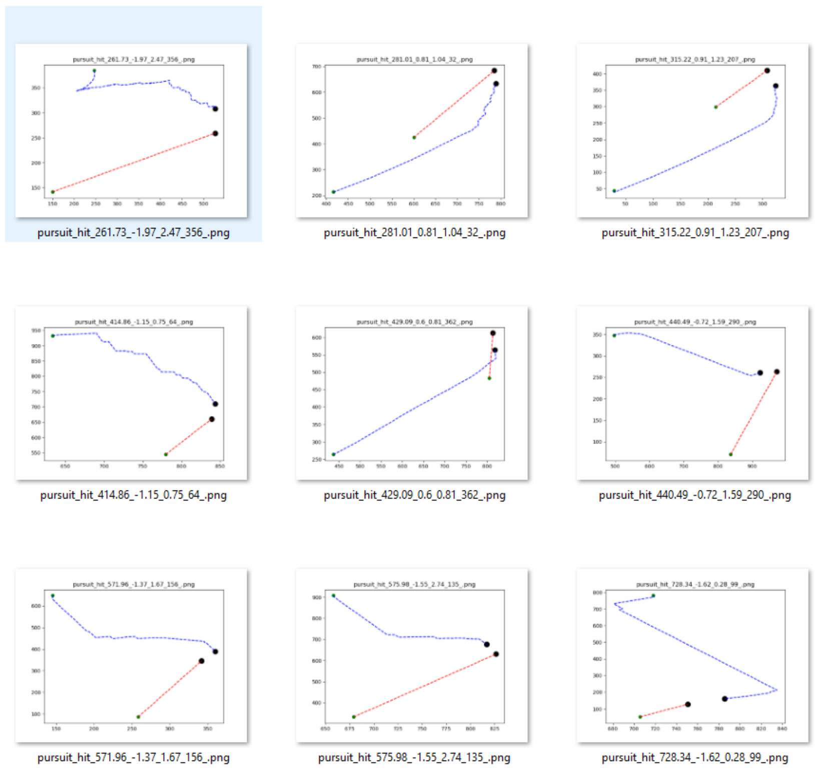


Figure 72 Strategy 6 Game Samples

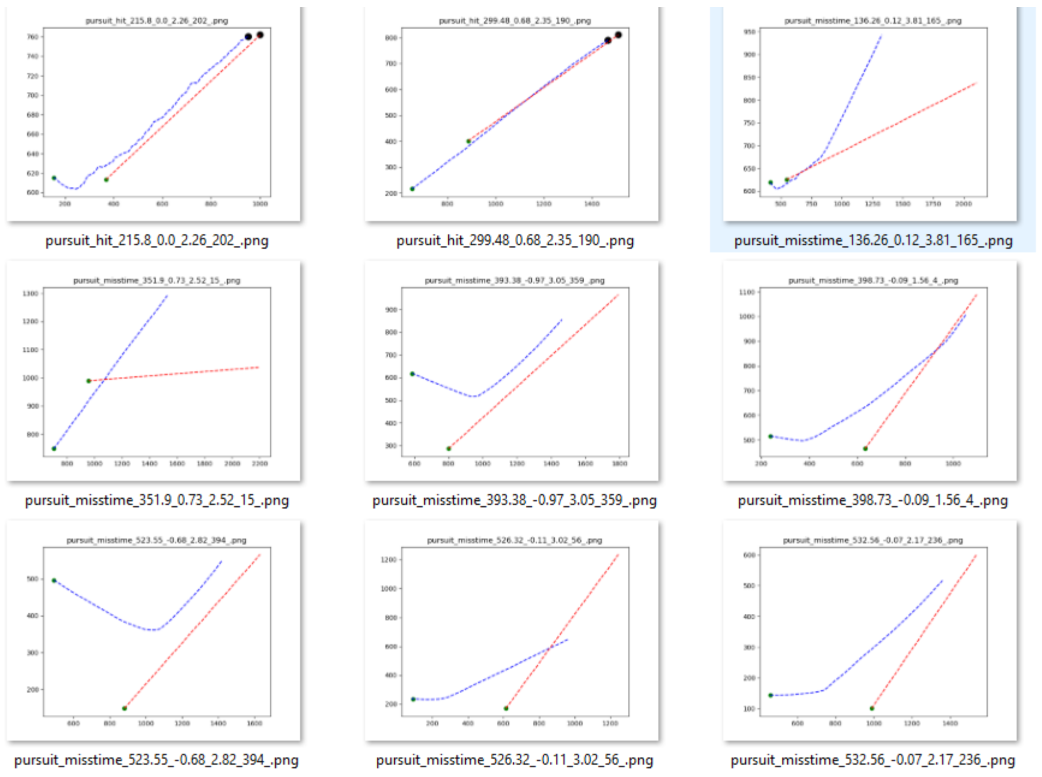


Figure 73 Strategy 7 Game Samples

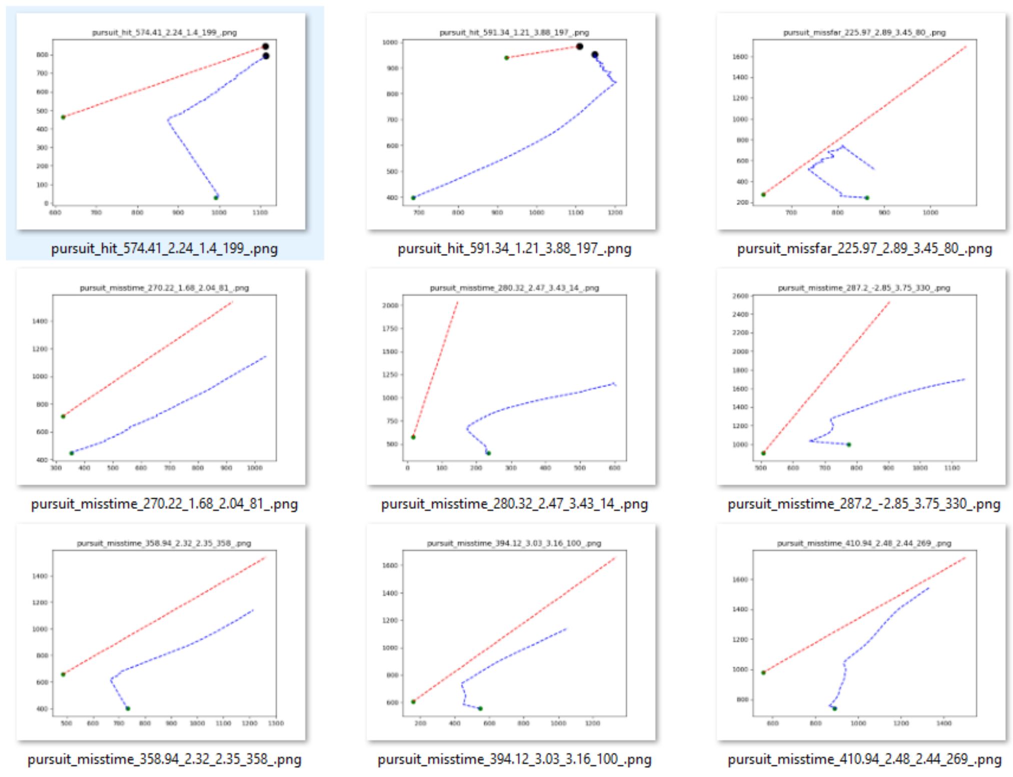


Figure 74 Strategy 8 Game Samples

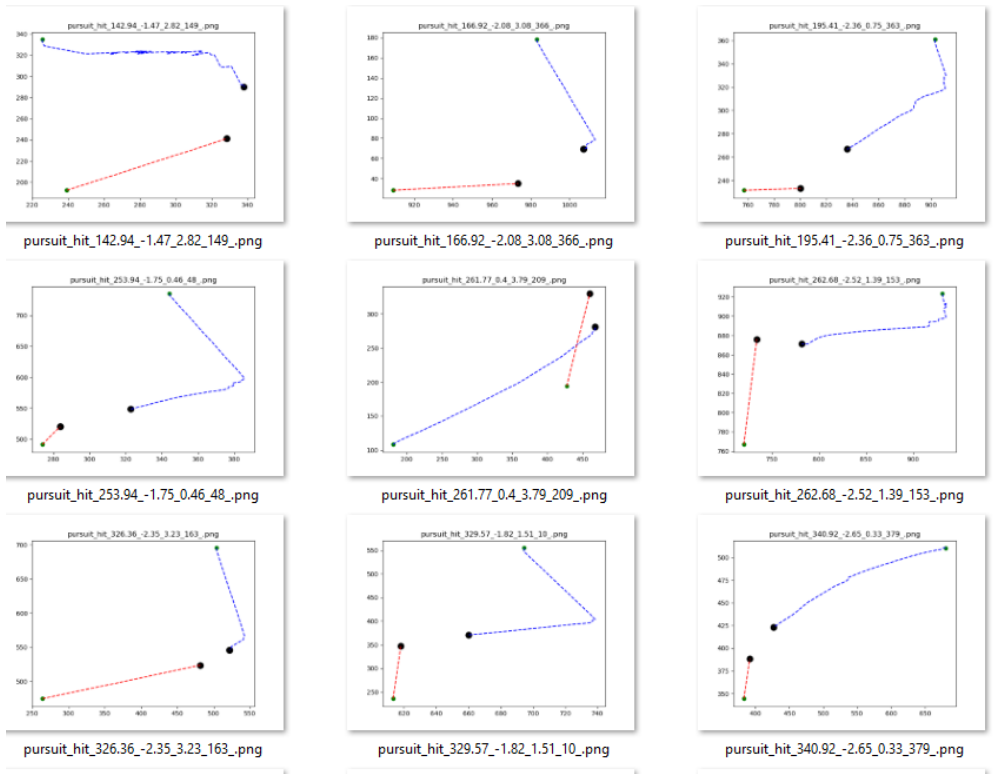


Figure 75 Strategy 9 Game Samples

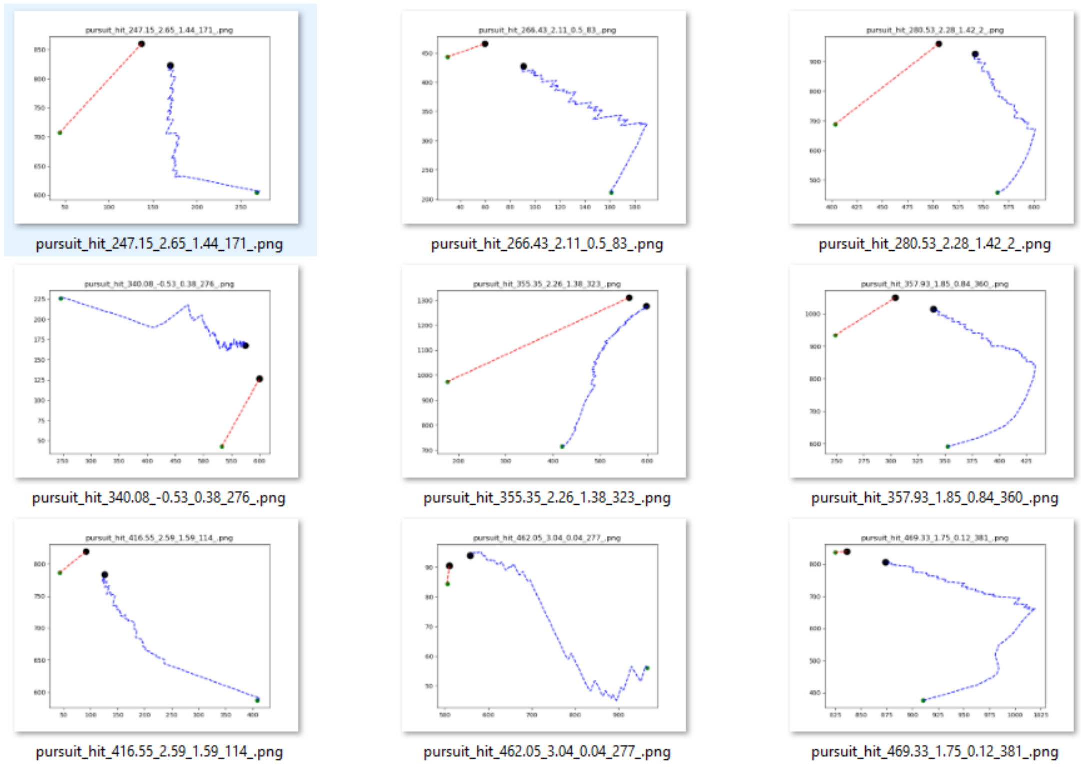


Figure 76 Strategy 10 Game Samples

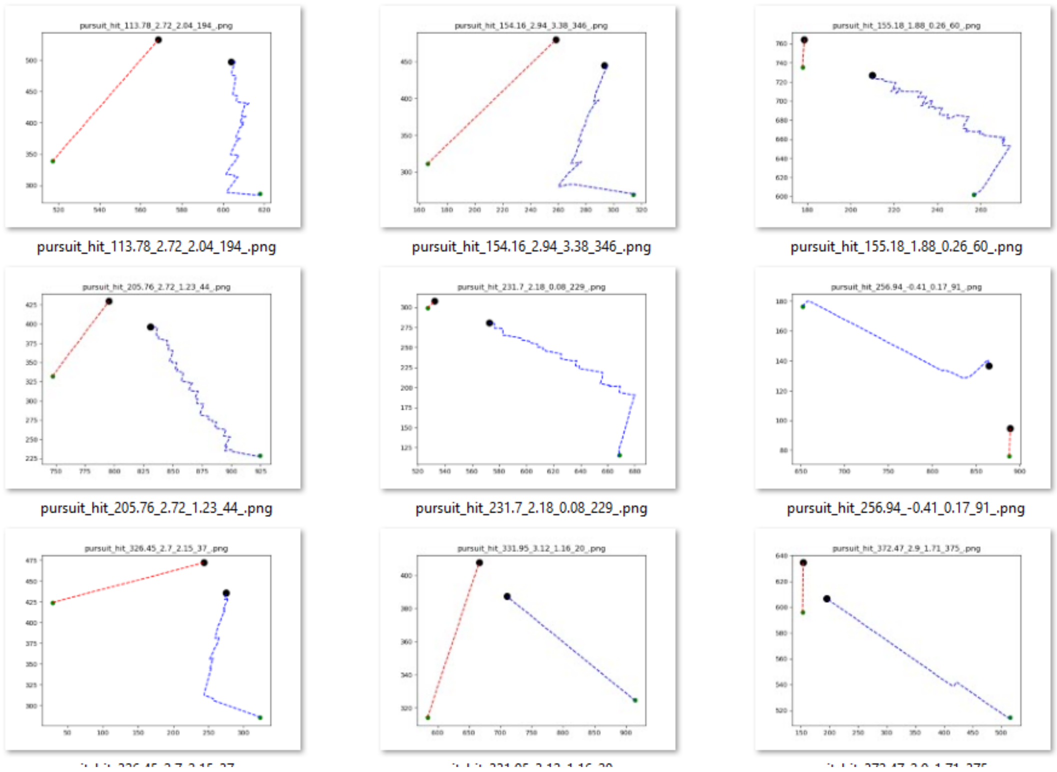


Figure 77 Strategy 11 Game Samples

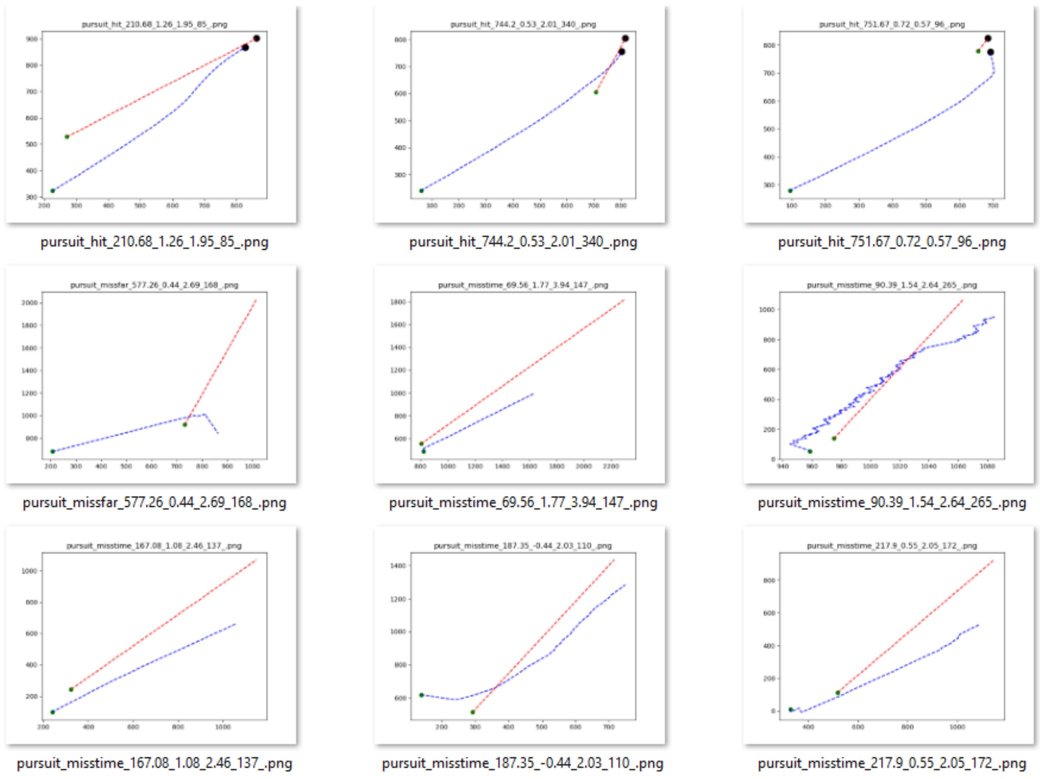


Figure 78 Strategy 12 Game Samples

<u>Section 1: Analysis of dimension-reduced actions (heading & acceleration only)</u>

In Figure 79, we show the same conditions studied for behavior analysis as they relate to the UMAP-embedded *action* space (Chapter 2 Section 2.3). Again, we see by visual inspection that the dimension-reduced *behavior (activation pattern)* space (Figure 21) separates better over the conditions than those of the action space shown here. Because of this, we can better estimate machine strategies from conditions by analyzing machine *behaviors* defined as activation patterns rather than their associated *actions*. That is, because of the lack of separation along these conditions, we cannot draw conclusions like "favorable initial angles to the target lead to Strategy 3: 'head-on approach'" as we can do when we examine relationships between conditions and strategies in the dimension-reduced *behavior (activation pattern)* space (Figure 21).

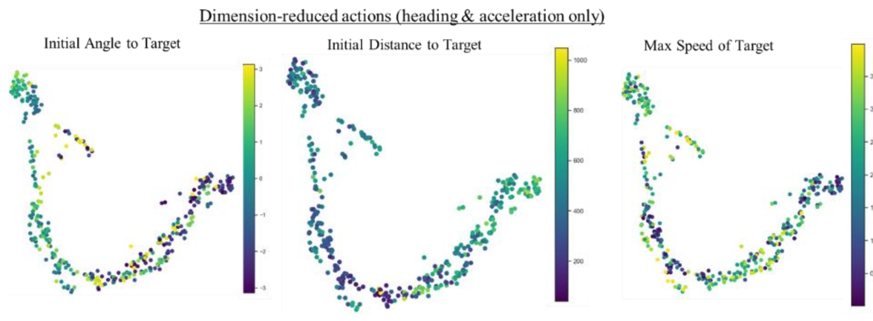


Figure 79 UMAP-embedded actions have less informative separation over user-proposed game conditions.

Moreover, in Figure 80, we see that clustering with respect to actions to determine strategies also leads to less separation over game outcomes. That is, every strategy leads to a heterogeneous mixture of game outcomes. In other words, there are no strategies in the bar plot that contain only one color (game outcome). This again motivates the analysis of underlying activation patterns more so than actions alone to glean useful relationships between strategies and performance.

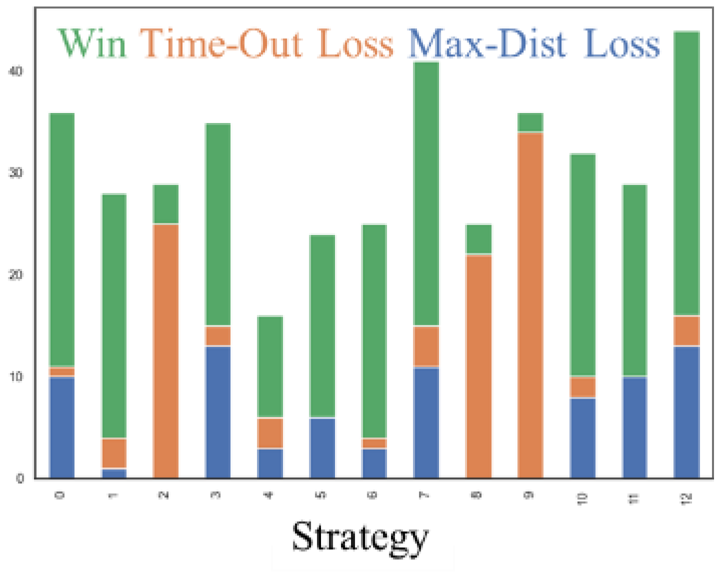


Figure 80 Action-clustered strategies separate less homogeneously over game outcomes.

The associated strategies for the action space are provided in Figure 81. The games associated with each of the clusters were not analyzed for human interpretability.

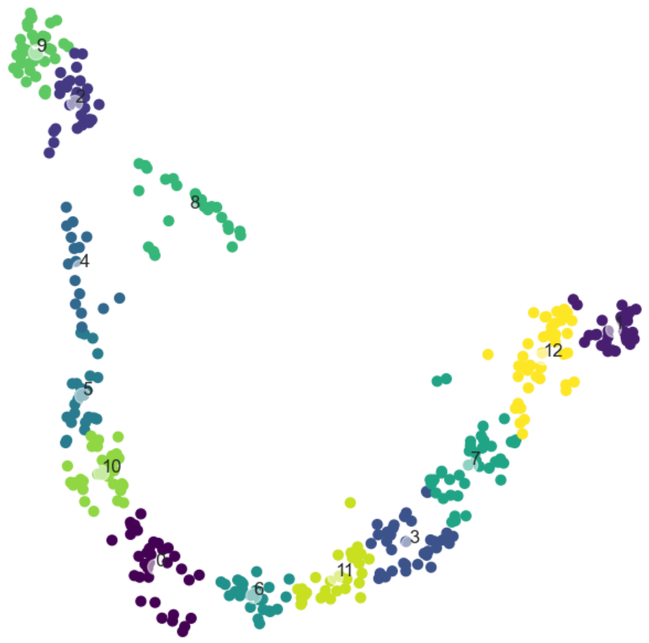


Figure 81 Strategy clusters for UMAP-embedded actions

Bibliography

References

[1] A. Barredo Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion,* pp. 82-115, 2020.

[2] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *PNAS,* vol. 116, no. 44, pp. 22071-22080, 2019.

[3] E. Schmidt, R. Work, S. Catz, E. Horvitz, S. Chien, A. Jassy, M. Clyburn, G. Louie, C. Darby, W. Mark, K. Ford, J. Matheny, J.-M. Griffiths, K. McFarland and A. Moore, "Final Report, Chapter 7: Establishing Justified Confidence in AI Systems," The National Security Commission on Artificial Intelligence, 2021.

[4] D. Gunning and D. Aha, "DARPA's Explainable Artificial Intelligence (XAI) Program," *AI Magazine,* pp. 44-58, Summer 2019.

[5] R. Isaacs, "Games of Pursuit," RAND Corporation, 1951.

[6] P. Cheng, "A Short Survey on Pursuit-Evasion Games," 2003.

[7] T. H. Chung, G. A. Hollinger and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robotics,* vol. 31, no. 299, 2011.

[8] A. Merz, "The homicidal chauffeur - a differential game: PhD thesis," *Stanford Univ.,* 1971.

[9] Y. Feng, L. Dai, J. Gao and G. Cheng, "Uncertain pursuit-evasion game," *Soft Comput,* vol. 24, p. 2425–2429, 2018.

[10] K. Quigley, S. A. Gabriel and S. Azarm, "Multi-Agent Unmanned Vehicle Trajectories with Rolling-Horizon Games," *Military Operations Research Society Journal,* 2020.

[11] M. Wang, L. Wang and T. Yue, "An Application of Continuous Deep Reinforcement Learning Approach to Pursuit-Evasion Differential Game," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC),* Chengdu, China, 2019.

[12] A. A. Al-Talabi and H. M. Schwartz, "A two stage learning technique using PSO-based FLC and QFIS for the pursuit evasion differential game," in *2014 IEEE International Conference on Mechatronics and Automation*, Tianjin, 2014.

[13] S. F. Desouky and H. M. Schwartz, "A novel technique to design a fuzzy logic controller using Q-learning and genetic algorithms in the pursuit-evasion game," in *2009 IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, TX, 2009.

[14] A. T. Bilgin and E. Kadioglu-Urtis, "Ana pproach to multi-agent pursuit evasion games using reinforcement learning," in *2015 International Conference on Advanced Robotics (ICAR)*, Istanbul, 2015.

[15] Google, "Explainable AI," [Online]. Available: https://cloud.google.com/explainable-ai.

[16] IBM, "Explainable AI," 2021. [Online]. Available: https://www.ibm.com/watson/explainable-ai. [Accessed 23 April 2021].

[17] P. Krishnamurthy, F. Khorrami, S. Schmidt and K. Wright, "Machine Learning for NetFlow Anomaly Detection with Human-Readable Annotations," *IEEE Transactions on Network and Service Management*, 2021.

[18] S. Schmidt, J. Stankowicz, J. Carmack and S. Kuzdeba, "RiftNeXt(TM): Explainable Deepn Neural RF Scene Classification," in *14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, Abu Dhabi, United Arab Emirates, 2021.

[19] M. Sundararajan and A. Najmi, "The many Shapley values for model explanation," *preprint arXiv,* 2019.

[20] J. Hilton, N. Cammarata, S. Carter, G. Goh and C. Olah, "Understanding RL Vision," *Distill,* 2020.

[21] L. Schubert, M. Petrov, S. Carter, N. Cammarata, G. Goh and C. Olah, "OpenAI Microscope," 14 April 2020. [Online]. Available: https://openai.com/blog/microscope/. [Accessed 21 April 2021].

[22] S. Booth, Y. Zhou, A. Shah and J. Shah, "BAYES-TREX: a Bayesian Sampling Approach to Model Transparency By Example," *Association for the Advancement of Artificial Intelligence,* 2021.

[23] G. Vilone and L. Longo, "Explainable Artificial Intelligence: a Systematic Review," *preprint arXiv,* 2020.

[24] C. Firestone, "Performance vs. competence in human–machine comparisons," *Proceedings of the National Academy of Sciences of the United States of America (PNAS),* 2020.

[25] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. Garcia Castaneda, C. Beattie, N. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu and T. Graepel, "Human-level performance in 3D multiplayer games with population-based reinforcement learning," *Science,* pp. 859-865, 2019.

[26] T. Zahavy and S. Mannor, "Graying the black box: Understanding DQNs," in *International Conference on Machine Learning*, New York City, New York, 2016.

[27] P. Rauber, S. Fadel and A. Falcao, "Visualizing the hidden activity of artificial neural networks," *IEEE transactions on visualization and computer graphics,* vol. 23, no. 1, pp. 101-110, 2017.

[28] M. Ali, M. W. Jones, X. Xie and M. Williams, "TimeCluster: dimension reduction applied to temporal data for visual analytics," *The Visual Computer,* pp. 1013-1026, 2019.

[29] L. McInnes and J. Healy, "Uniform Manifold Approximation and Projection for Dimension Reduction," *preprint ArXiv,* 2018.

[30] T. P. Lillicrap, J. J. Hunt and A. Pritzel, "Continuous Control with Deep Reinforcement Learning," *arXiv preprint,* 2015.

[31] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine,* vol. 19, no. 1, pp. 5-18, 2004.

[32] Y. Kim and H. Bang, "Introduction and Implementations of the Kalman Filter," IntechOpen, 2018. [Online]. Available: https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter-and-its-applications. [Accessed 24 April 2021].

[33] M. S. Grewal and A. P. Andrews, "Applications of Kalman Filtering," *IEEE Control Systems Magazine,* 2010.

[34] D. Silver, J. Schrittwieser and K. Simonyan, "Mastering the Game of Go without Human Knowledge," *Nature,* vol. 550, pp. 354-359, 2017.

[35] D. Silver, G. Lever, N. Heess and T. Degris, "Deterministic policy gradient algorithms," in *ICML,* 2014.

[36] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR,* 2015.

[37] J. F. Allen, S. Schmidt and S. A. Gabriel, "Reinforcement Learning Approach to Speed-Overmatched Pursuit Games with Uncertain Target Information," in *Naval Applications of Machine Learning (NAML)*, Virtual, 2021.

[38] N. Bharti, "What-is-meant-by-activation-function," Quora, 23 October 2018. [Online]. Available: https://www.quora.com/What-is-meant-by-activation-function. [Accessed 4 June 2021].

[39] M. Hazewinkel, "Encyclopedia of Mathematics," EMS Press, 1994.

[40] C. T. Dodson and T. Poston, Tensor geometry. Graduate Texts in Mathematics, Berlin, New York: Springer-Verlag, 1991.

[41] W. Dong, C. Moses and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *20th International Conference on World Wide Web*, New York, NY, 2011.

[42] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits," Courant Institute, NYU Corinna Cortes, Google Labs, New York Chirstopher J C Burges, Microsoft Research, Redmond.

[43] R. L. Thorndike, "Who Belongs in the Family?," *Psychometrika,* vol. 18, no. 4, pp. 267-276, 1953.

[44] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Russwurm, K. Kolar and E. Woods, "Tslearn, A Machine Learning Toolkit for Time Series Data," *Journal of Machine Learning Research,* vol. 21, no. 118, pp. 1-6, 2020.

[45] Defense Advanced Research Projects Agency (DARPA), "Competency-Aware Machine Learning (CAML)," 2019. [Online]. Available:

https://www.darpa.mil/program/competency-aware-machine-learning. [Accessed 23 April 2021].

[46] A. Klyubin, D. Polani and C. L. Nehaniv, "All else being equal be empowered," in *European Conference on Artificial Life*, Berlin, Heidelberg, 2005.

[47] A. S. Klyubin, D. Polani and C. L. Nehaniv, "Keep your options open: An information-based driving principle for sensorimotor systems," *PloS one,* vol. 3, no. 12, p. 4018, 2008.

[48] T. Jung, D. Polani and P. Stone, "Empowerment for continuous agent - environmental systems," *Adaptive Behavior,* vol. 19, no. 1, pp. 16-39, 2011.

[49] D. Pathak, P. Agrawal, A. Efros and T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," in *Proceedings of the 34th International Conference on Machine Learning (PMLR)*, 2017.

[50] S. Dey, K.-W. Huang, P. A. B eerel and K. M. Chugg, "Characterizing Sparse Connectivity Patterns in Neural Networks," in *Information Theory and Applications Workshop (ITA)*, San Diego, CA, 2018.

[51] T. M. Book and J. A. Thomas, Elements of Information Theory, Hoboken, NJ: Wiley, 1991.

[52] T. S. Community, "scipy.stats.entropy," SciPy.org, [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.entropy.html. [Accessed 5 June 2021].

[53] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal,* vol. 27, no. 3, pp. 379-423, 1948.

[54] W. H. Kruskal, "Historical Notes on the Wilcoxon Unpaired Two-Sample Test," *Journal of the American Statistical Association,* vol. 52, no. 279, pp. 356-360, 1957.

[55] C. Marais, "Empowerment as Intrinsic Motivation," towards data science, 18 July 2018. [Online]. Available: https://towardsdatascience.com/empowerment-as-intrinsic-motivation-b84af36d5616. [Accessed 24 April 2021].

[56] T. Campbell, J. Straub, J. W. Fisher III and J. How, "Streaming, Distributed Variational inference for Bayesian Nonparametrics," *Advances in Neural Information Processing Systems,* vol. 28, pp. 280-288, 2015.

[57] J. Chang and J. W. Fisher III, "Parallel Sampling of DP Mixture Models using Sub-clusters Splits," *Neural Information and Processing Systems,* 2013.

[58] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[59] Y. W. Teh, M. I. Jordan, M. J. Beal and D. M. Blei, "Hierarchical Dirichlet Processes," *Journal of the American Statistical Association,* vol. 101, no. 476, pp. 1566-1581, 2006.

[60] D. Aldous, "Exchangeability and Related Topics," *Ecole d'Ete de Probabilities de Saint-Flour XIII,* pp. 1-198, 1983.

[61] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision*, 2011.

[62] C. Molnar, Interpretable Machine Learning, github, 2019.

[63] "Additional information is available as supplementary materials.".

[64] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[65] R. K. Pathria and P. Beale, Statistical Mechanics (Third ed.), Academic Press, 2011.