The Use of AI/ML to Replicate Threat Behaviors for Nonlinear Simulation

Charles Etheredge, Kyle Russell, William Marx, CAPT Timothy Hill, USN (ret),
Col (ret) Daron Drown, USAF
Intuitive Research and Technology Corporation
Internal Research and Development
Huntsville, AL
chad.etheredge@irtc-hq.com; kyle.russell@irtc-hq.com; william.marx@irtc-hq.com;
timonthy.hill@irtc-hq.com; daron.drown@irtc-hq.com

ABSTRACT

The development of models that replicate complex, non-linear, adaptive, constructive threats poses challenges for training simulation developers. As threats to our nation's defensive systems mature, our ability to characterize and model them needs to improve so that we are not outmatched. This paper reports on the progress of our Research and Development (R&D) implementing interdisciplinary methods for Big Data analytics, Artificial Intelligence (AI), and Machine Learning (ML) to capture, characterize, and replicate red player/threat behaviors. We used a progressively complex development and testing framework. First, we selected a pathfinder application that was sufficiently complex to require game play and adaptive behavior on both sides. The paper will discuss our implementation of a Mini-max algorithm, a recursive algorithm used in decision-making, and our rationale for using this algorithm to benchmark the progress of our ML algorithms, with comparisons to others' implementations cited in relevant research. As we tested our AI against humans, we tailored several AI parameters to make the system easier or more challenging as a human opponent. To capture human behaviors, we record all moves made by both sides, per game, and per event. An event is defined as a discrete period, usually 3 days, where multiple human players' moves are captured. Our solution to this challenge required us to train our ML algorithms on the individual and collective behavior of human players at an event. The paper will describe how we have been able to replicate human player/threat behavior for three events, with references to others' implementation in scientific literature. The paper will also describe logical extensions to problems of interest to the Department of Defense, Department of Homeland Security, and cyber security. Those domains include human, machine, and cyber players and opponents whose behavior can be replicated using our AI/ML methods.

ABOUT THE AUTHORS

Mr. Charles Etheredge is a Software Engineer with Intuitive Research and Technology Corporation. As a member of the research and development team at Intuitive, he is tasked with performing and demonstrating research for the company. He has bachelor's degrees in both, computer science and business administration from the University of North Alabama. Currently, he is pursuing a Master's in Machine Learning and Artificial Intelligence at Georgia Tech. He has expertise with video and metadata encoding solutions, as well as web, cloud, and AI applications. Additionally, he has participated in a wide variety of research, including data visualization, image analysis, cryptography, and AI weather forecasting.

Mr. Kyle Russell is a Senior Digital Engineer with Intuitive Research and Technology Corporation. As a member of the research and development team he is responsible for exploring new applications of cutting-edge technologies to customer problems. He has a bachelor's degree in Electrical Engineering from the University of Alabama and is currently pursuing a Master's degree in Computer Science with focuses in Artificial Intelligence/Machine Learning and Data Analytics. He has experience developing advanced real-time interactive visualization solutions, and Big Data Analytics pipelines, as well as experience with modern web technologies and database systems in general.

Dr. William Marx is the Senior Vice President and Chief Technology Officer of Intuitive Research and Technology Corporation in Huntsville, Alabama. He is responsible for planning, managing, and executing research and development programs aligned with the technology priorities of the US military and commercial customers. His experience base and technical portfolio include advanced visualization systems, Big Data analytics, Artificial Intelligence and Machine Learning, Knowledge Based Systems, missile system design, multi-disciplinary design optimization, missile guidance and control, analysis of exo-atmospheric kill vehicles, supersonic aircraft design, and

ground and space robotic system design. Dr. Marx received his PhD and MS in Aerospace Engineering from the Georgia Institute of Technology and his BS in Aerospace Engineering with a minor in Mathematics from Embry-Riddle Aeronautical University. He was a NASA Langley GSRP Fellow.

Mr. Timothy Hill is Director of Central Florida Operations with Intuitive Research and Technology Corporation's Orlando office. He is responsible for efficient operation of the Central Florida office and for representing *INTUITIVE* in Central Florida with all customer sets and industry and academic teammates. He is a retired Navy Officer who served across a wide range of operational, staff, and acquisition assignments, culminating in his final tour commanding the Naval Air Warfare Center Training Systems Division (NAWCTSD). He amassed over 3,200 flying hours and 700 carrier arrested landings in more than 32 aircraft types, including operational assignments in the S-3B Viking and F/A-18F Super Hornet. He earned a BS in Systems Engineering from the U.S. Naval Academy, a MS in Systems Engineering from Johns Hopkins University, and a MS in International Relations from Troy University. He is a graduate of the U.S. Naval Test Pilot School and the Air Command and Staff College. He is an active member of the Central Florida community through board service and other similar activities.

Mr. Daron Drown is a technical program manager and test and evaluation and autonomy SME at Intuitive Research and Technology Corporation's Huntsville headquarters. He is a retired Air Force Test Pilot with specific expertise in fifth generation combat aircraft, radar and optical sensors, guided munitions, AI-driven autonomous battle management, mission planning, and missile defense. He also has significant experience operating advanced military simulators and training systems, both as student and instructor. He earned a B.S. in Mechanical Engineering from the U.S. Air Force Academy, an M.S. in Engineering Mechanics from Michigan State University, and M.S. in Systems Engineering from the Air Force Inst. of Technology, and an M.S. in National Resource Strategy from National Defense University.

INTRODUCTION

Modeling and Simulation (M&S) plays a key role in the design, development, test, and operation of modern military aircraft and missile systems. The spectrum and application of M&S are both broad and deep, ranging from detailed physics-based, engineering simulations of individual components to statistics-based system-of-system level engagement simulations. This begins with high-level concept development where the effects of a new weapon, platform, or capability are modeled in context of the integrated environment, both friendly and adversary, within which the new item will operate. As development continues, military testers must demonstrate actual and simulated performance of our weapon systems against credible, validated threat systems and models. There are many ways to build threat representations – ranging from simple statistics-based models that perform according to pre-determined metrics, to physics-based models and simulations that mimic the characteristics and behaviors of diverse airborne, space-based, and ground-based platforms and vehicles. Increasingly sophisticated threat models are needed for operational testing – the nature of our modern system-of-systems capabilities and our adversaries' increasingly complex offensive systems makes ground and air testing prohibitively expensive. Our military has been pursuing and developing Live, Virtual, and Constructive (LVC) training for many years due to its advantages over legacy digital-only or physical-only training and evaluation methods. The USAF Distributed Mission Operations (DMO) environment has successfully trained hundreds of aircrews effectively, and the Navy and Marine Corps are actively using and improving similar capabilities. These systems have been cross-linked successfully, and the Services are pursuing capability for persistent, on-demand linkages for these systems at all classification levels. Similarly, much has been reported about the Joint Simulation Environment (JSE), the latest effort to build an adequate environment to fully test the operational effectiveness of the F-35, while also providing a robust environment for tactics and future capability development. These initiatives have brought about M&S products such as the Integrated Threat Analysis Simulation Environment (ITASE) and the Next Generation Threat System (NGTS) that seek to deliver validated threat models at the appropriate fidelity for the intended purpose. However, there is an increasing difficulty in replicating high-end enemy capabilities – such as advanced fighters and surface-to-air missiles – with live "red" teams. (Harper, 2015). The challenges include cost of replication, physical limitations of large-scale modern, high-end warfare scenarios, and technical hurdles associated with *imitating the non-linear behaviors* of live red team adversaries.

Ideally, complex threats react to friendly tactics, techniques, and procedures (TTPs) and countermeasures in accordance with their capability and established national doctrine. The modeling of threats with performance and/or physics-based models that accurately represent the assessed kinematic capabilities of adversarial red team assets is not enough. Threat systems may be networked in communication that enhances their overall capability; they may be maneuvering or employing countermeasures; and they should be responsive to friendly force actions. LVC methods can provide live actors for red roles, but the actors need to be trained to correctly mimic the behaviors and doctrine of the red assets. That training takes time, slowing our ability to "digest" new information in development efforts and to be prepared through training. Myriad DoD initiatives have addressed various elements of generating, validating, distributing, and integrating threat models into our nation's portfolio of test and evaluation simulations and architectures. Gaming methods have been a part of weapon system development for a long time. In recent years, Serious Games and Serious Gaming methods have found increasing application within the DoD M&S community. While war games are used to stimulate decision making in the conduct of military operations and planning, the primary focus of serious games is to support education and training. (Hartman & Allen, 2017). This has been extended to LVC methods using environments such as Virtual Battlespace Simulation (VBS) (Pollack, 2012). There are opportunities to use gameplay concepts and artificial intelligence and machine learning (AI/ML) to bridge the gaps between wargaming for decision making and M&S for performance evaluation.

Developers have near-perfect understanding of both the performance characteristics and TTPs associated with friendly systems, along with access to learning that occurs throughout the life cycle. Compounding the threat modeling problem, we are constantly learning about the performance characteristics for threat systems, and our knowledge is not usually perfect. Traditionally, learning about threat systems performance characteristics is incorporated into validated updates to threat models, while updated understanding of TTP and doctrine is provided to adversary forces in the form of updated training. These processes take time, and while they are being accomplished, decision makers continue to take action with stale information and warfighters are undertaking fundamentally negative training. These gaps need to be closed.

As threat technologies advance, are more robustly integrated, and TTPs/doctrine become more complex, the rate of change associated with these characteristics is also increasing. As a result, our modeling of modern adaptive threats needs to advance. We can use modern AI/ML methods to create data sets that can be used to train and replicate non-linear, adaptive threat behaviors, such as human decision making in executing doctrine or the behavior of highly networked threat systems. These models can then be used within our M&S community to make better choices in pursuing and developing new capabilities and to better train our warfighters, ensuring that our nation's defenses are ready to stand against the full spectrum of modern threats – whether airborne, space-based, ground-based, human, or digital. This paper presents methods we are developing and using to capture, analyze, characterize, and then mimic non-linear threat behavior.

LVC environments with non-linear AI-developed threat models must be architected with modularity such that any specific scenario feature can be flexibly swapped between Live, Virtual, or Constructive environments. AI-developed models can operate stand-alone in Constructive use, or as augmentation to humans in Virtual environments (combining elements of artificial intelligence and human intelligence in real-time, or as AI-agents operating in live Manned-Unmanned Teaming (MUMT) arrangements (Fawkes & Menzel, 2018).

PRIOR EXAMINATIONS ON MODELING OF THREAT BEHAVIOR

Modeling threat behavior is a complex problem and one that many industries are trying to accomplish. In the DoD domain, simulating threats is typically a very thorough but often rigid approach to modeling real world assets. These simulations vary between statistics-based models and physics-based models. These models are often thoroughly verified and validated. Therefore, it can be difficult to quickly adapt these models to accommodate new threat behaviors. This area can greatly benefit from the inclusion of AI and ML technologies that can more quickly incorporate, learn, and act on new information. Continuous Learning (CL) is one of the many benefits that ML technologies offer. In a CL ML algorithm, the algorithm constantly adapts and learns from new information included in its learning database. With the inclusion of this technology, the modeling and simulation realm can respond more quickly to the new information including non-linear threat behaviors.

In the commercial gaming industry, the concept of a Non-Player Character (NPC) is an interesting and expanding concept. The NPC is essentially an AI which is trained to either aid or threaten the human players in the game.

NPC's or similar unmanned actors are typically referred to as constructive entities in DoD applications, and for the purposes of this paper, NPC's and constructive entities will be considered synonymous. Games employing NPCs can vary widely in disciplines such as educational games, role-playing games, first-person shooters, etc. (Andhik et. Al, 2019) examined the use of NPCs in educational games and found that the inclusion of NPCs increased human user interest. In 2016, it was proposed that many of the problems with NPCs stem from their lack of believability (Warpefelt). Warpefelt asserts that this lack of believability lessens the enjoyment of the game for the user. This is even more true in DoD applications where realism is lost, rather than merely experiencing a reduced entertainment value. For that reason, while these studies are tailored towards the commercial gaming industry, the findings can be applied to linear and non-linear adversarial threat modeling. AI algorithms can be taught how to perform certain tasks. However, an AI thinks and performs very differently from humans. AI actions and decisions are very calculated and are not based on social norms. If these AI are meant to replicate human behavior such as simulating war gaming, then their actions may not reflect current threat vectors that are more in line with how human threats would behave. Therefore, believability and behavior modeling are very important factors when creating an AI that simulates a dynamic, adaptive threat landscape.

AI/ML FOR BEHAVIOR MODELING

The use of AI/ML for modeling and countering threat behavior is not a new concept. The earliest examples of this began in the 1950's with the usage of AI to play board games such as checkers, and later chess, at a competent level against human opponents (Copeland, 2020). Board games provide a simple starting point for understanding the growth in game complexity when modeling the behaviors a threat may take against you, as board games are constrained by a well-defined set of rules and a clear end goal. Looking at a simple game such as tic-tac-toe, there are a simple set of rules and a finite number of 3x3 board states that can be reached in a given game. Ignoring illegal and duplicate states, there are only 765 to be exact. With such few possible game states, it is possible to search through all possible future states each turn to find the current best path to victory. However, as games become more complex, searching all game states becomes impractical or infeasible. Consider a slightly more complicated game, Connect 4; the number of possible game states for this game increases to 4,531,985,219,092. Searching all possible states in a reasonable amount of time is simply not feasible.

This drastic increase in total game states is due to the increased problem state space and the increasing average branching factor migrating from Tic-Tac-Toe to Connect 4. While the increased number of states from the increased board size is a given, the increased branching factor is the primary factor in the increased number of possible states. For Tic-Tac-Toe, this average branching factor is approximately 4. The average branching factor for Connect 4 is approximately 7. This factor describes the average number of states that can be reached from a given state. The issues with state-based approaches become apparent as the average branching factor increases. Moving on to even more complex games, like chess, the branching factor increases to 35 and the total number of possible game states rises to a lower bound of 10^{50} and an upper bound of 10^{120}. At this point, searching all states is not only time consuming, but very memory intensive. As such, methods are needed to reduce the problem space; MiniMax and AlphaBeta pruning are two very well know algorithms for this use case.

MiniMax is a recursive algorithm for minimizing the possible loss and maximizing the possible gain in any scenario in which states can be both represented in memory and scored for some value reflecting the benefit or detriment of reaching that state. MiniMax, however, does not reduce the total number of states that need to be searched, but it does provide a method for determining the value of any give state that is exactly what is needed for effective use of the AlphaBeta pruning algorithm.

AlphaBeta pruning provides a method for *pruning* entire portions of a search space, removing the need to search the pruned states. States that are pruned are those that lead the maximizing player to detrimental situations. The general idea of AlphaBeta being that: if a move appears to be bad then do not take the time to determine exactly how bad it is. This is much more in line with how humans approach similar problems; when playing a game, we recognize poor move choices, ignore them, and focus on those that bring us closer to the goal. The benefit of using these two algorithms for efficient problem space searches is abundantly clear; using Connect 4 as an example, if we want to search all states five turns ahead, then approximately 16,807 states would need to be evaluated, but using our coupled MiniMax and AlphaBeta implementation, the number of states that need to be searched is reduced to approximately 3,000.

Beyond solving state-based problems with algorithmic approaches, Machine Learning (ML) has also proven to be highly effective for playing games at an expert level. Algorithmic approaches, while effective and capable of producing *best* choices, have a glaring weakness, that is the time required to determine that choice as game complexity increases. ML offers a solution to this weakness. Through the use of ML concepts, a program can *learn* how to play a game by training on data from real players. One such ML approach is reinforcement learning. Reinforcement learning is a paradigm of ML focused on the use of intelligent agents and the ability to maximize some cumulative reward function given by the actions of those agents. This differs from supervised and unsupervised ML approaches, as reinforcement learning does not require labeled input or output data for training. As such, reinforcement learning agents can be trained using sub-optimal data, in that the training data does not need to contain only explicitly correct actions to be learned. Through the use of reward functions the agent, in a sense, is able to optimize itself. Notably, this does not produce the optimal solutions that can be achieved with algorithmic approaches, but it does produce nearly-optimal solutions. Additionally, once a model is trained, it is able to produce results much faster than an algorithmic approach.

Another advantage of using an ML approach is that it removes the need to account for all of the rules and niche situations that come with manually writing a program for a game. If the data sufficiently covers all use cases, the model should be able to learn how to respond to all types of situations on its own. This allows for ML approaches to be applied to a much broader array of problems, as it removes the issues which arise from increasing complexity. While ML can be used to solve many problems, it is not without its disadvantages. There are many methods for implementing and training ML models, each with unique challenges and limitations.

OUR APPROACH

Our goal was to create an ML agent capable of learning and imitating the behaviors of players in a game, without the need to provide the rules of the game or define types of behaviors. The ML agent should be able comprehend player behaviors and the rules of the game based solely on data gathered from real people playing the game. Notably, our primary goal was not to create an ML model which could out perform and beat the algorithmic approaches, but to create an ML model that could imitate the playstyle of real people who had played our game at various conferences and events, as the people playing would not be experts at the game. We choose Connect 4 to test this process, as it is complex enough to allow for unique and discernable player behaviors, while being simple enough that the majority of people both know how to play and would be able to complete a game quickly. Further, to increase the odds of players engaging with our game at conferences, we created iPad application for our game using Unreal Engine 4. This proved highly effective, as it made the game more approachable and allowed us to show people analytics of their games in real-time. The UI for our Connect 4 game is shown in the left half of Figure 1.

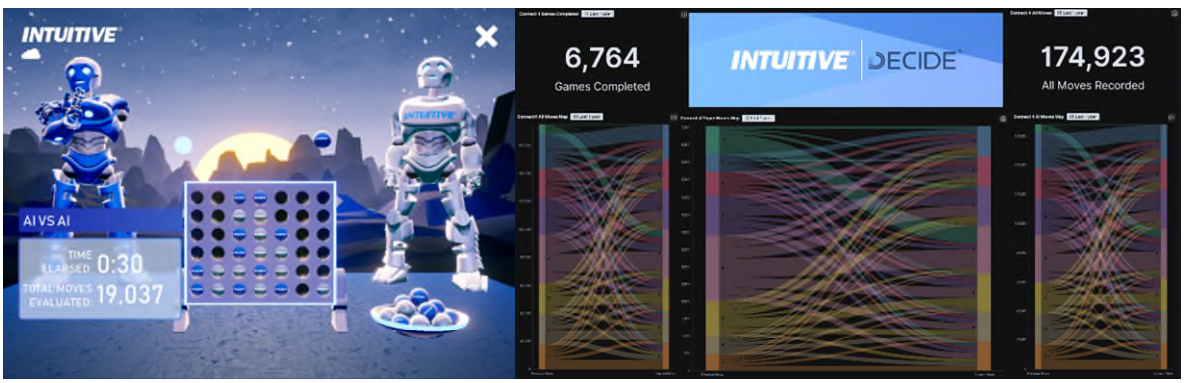

Fig. 1: QuadConnect iPad application created in Unreal Engine 4 (left) and real-time analytics dashboard using *INTUITIVE's* DECIDE platform (right). The dashboard updates in real-time as people play the game.

To gather data for training our model, the team built a cloud-based server to host the AI, database, and analytics tools. The MiniMax AI was also built in a depth-limited manner, allowing us to adjust the game to varying levels of difficultly by limiting the depth that the AI is allowed to search. Search depth is essentially the number of turns the AI is allowed to *think* ahead. Our ability to actively adjust the search depth of the AI opponent made the game more

approachable by participants; many players did not want to play against an AI opponent that they thought could compute a perfect solution for each move.
During play the program would log and capture all game data. This data would be stored in a database where we would prepare, clean, and analyze the data. From this data we would determine player behavior patterns such as player move preferences given previous moves. Our data was captured at various conferences throughout 2021, including the Space and Missile Defense (SMD) Symposium held in Huntsville, Alabama, and the NDIA's Air Armament Symposium held in Fort Walton Beach, Florida. We also captured game play data at internal company demonstrations of our Connect 4 system.

In total, we captured data from over 6,000 games and more than 170,000 moves between August, 2021 and April, 2022. However, we began training and testing our model after our first event at the SMD Symposium in August, 2021. From these games we generated a Sankey diagram to serve as a move map outlining the human player move preferences at conferences, shown in the right half of Figure 1. The tables provided in Figure 2 display the same information, but in a numerical format.

Human Moves		**Previous Move**							
		-1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Current Move	**1**	0.109375	0.422770	0.168014	0.064162	0.047219	0.090141	0.106803	0.143939
	2	0.059896	0.147933	0.338724	0.065896	0.078174	0.110798	0.065106	0.089015
	3	0.057292	0.093546	0.110512	0.347977	0.271773	0.083568	0.147769	0.061553
	4	0.671875	0.063089	0.133872	0.281503	0.372508	0.100469	0.050475	0.074811
	5	0.031250	0.082669	0.111411	0.076301	0.076600	0.348357	0.133138	0.125000
	6	0.044271	0.085569	0.073675	0.098266	0.121721	0.151174	0.356986	0.140152
	7	0.026042	0.104423	0.063792	0.065896	0.032004	0.115493	0.139722	0.365530
MiniMax Moves		**Previous Move**							
		-1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Current Move	**1**	0.0	0.076346	0.071735	0.092810	0.114185	0.091935	0.067501	0.043564
	2	0.0	0.076346	0.154639	0.122601	0.137534	0.110798	0.171475	0.166997
	3	0.0	0.153374	0.134021	0.285019	0.169039	0.124948	0.169298	0.395380
	4	1.0	0.385140	0.308419	0.181323	0.179594	0.387380	0.354927	0.191419
	5	0.0	0.180641	0.144759	0.112289	0.157205	0.139156	0.097441	0.130033
	6	0.0	0.093388	0.109966	0.102836	0.131137	0.089010	0.090365	0.033663
	7	0.0	0.034765	0.076460	0.103122	0.111307	0.078980	0.048993	0.038944

Fig. 2: Summarized data from our Sankey diagrams showing player move preferences given a previous move (where -1 indicates the first move of a game). The cells highlighted in green indicate the predominant trend in player behavior.

These charts display the odds of a move being selected by a player given a previous move, with -1 representing the first move of a game. This data highlights a notable behavior difference between human players and the MiniMax algorithm. We can see that human players have a high tendency to select the same move as their opponent. The MiniMax algorithm strongly prefers column 4, but also tends to favor columns 3 and 5 with an additional bias to the left-hand side of the board. The bias of the MiniMax algorithm is explainable. Statistically, the center columns have a higher probability to generate winning positions. For example, on the first turn of a game, column 4 provides five potentially reachable winning positions, while all other first moves provide four or less. The tendency to favor the left-hand side of the board more heavily is due to the algorithm scoring move positions left to right and keeping the earlier move in cases where moves are found to be equal. What is not explainable is the behavior of human players and the tendency they have to select the same move as their opponent's previous move. This non-linear, non-algorithmic human behavior is what we will seek to imitate with machine learning.

While these behavior graphs provide us with a baseline to compare the behavior of our ML model to that of the player data it was trained against, we also need to model to play at a similar skill level to what was observed from human players. From here, we used the player data to train our ML model. To generate benchmarking data to assess the progress and effectiveness of our models, we constructed a test environment that allowed us to set either player in a game to any of our available algorithms or models and play a pre-determined number of games. The first of

these players is an opponent that places pieces at random. The second is a player that utilizes the MiniMax and AlphaBeta algorithms and can be used to produce an optimal move each turn. The MiniMax/AlphaBeta opponent can, again, be *depth limited*, allowing us to limit the number of turns that it is allowed to search ahead in the game, creating multiple *good*, but sub-optimal, opponents for our models to test against. With our tools in place, it was time to implement and test our models.

FIRST ML IMPLEMENTATION

Our first behavioral cloning implementation consisted of a simple feed-forward neural network implementation in PyTorch. The objective for the network was determining the win, loss, and tie potential for a move based on whose turn it was and the board state. The network was designed with a multi-class classification problem in mind. The final configuration consisted of an input layer with 44 features representing: current player, the current move, and a flattened version of the 6 x 7 board. Multiple configurations of hidden layers were tested, however deeper and wider networks tended to perform poorly in our tests in this configuration. The single hidden layer was thus decided to use 30 features and a rectified linear unit activation function. Finally, an output layer consisting of 30 input and 3 output nodes gave us the raw output for our network.

Training the network consisted of weighted random sampling based on label frequency to balance the dataset. We used cross entropy for our loss function and Adaptive Moment Estimation (Adam) as our optimizer after testing others and finding Adam to have the best performance. Further, we did not use a fixed number of training epochs, rather we trained until the loss converged to a stable threshold before stopping. When training the network, we opted to train based on a single event; meaning only data from a given event would be used for the training of a model. For example, we created models to clone the behavior of players at the Space and Missile Symposium. We created additional models to clone the behaviors of players at the Air Armament Symposium.

Use of the trained network was handled by exporting the PyTorch model into a format known as ONNX (Open Neural Network Exchange), is a high-performance package for pre-trained model inference. Additionally, because the network had been trained to evaluate proposed moves with board states, it was required to iterate over all possible valid moves for a given board state and evaluate the outcome. This would return win, loss, and tie percentages that were then evaluated to determine policy.

FIRST ML IMLPEMENTATION RESULTS

In testing our first ML model, we analyzed how closely the ML model matched our recorded player behavior and win rates. To do this, we ran the ML model for a total of 12,000 games. These games played the ML model against a random opponent and the ML model against the AlphaBeta AI at varying levels of difficultly, as this mimics the types of games the human players participated in.

We found that the win rate of this ML model was approximately 43.5%. This was promising, as it was close to that of the real player win rate of 38.8%. However, examining the actual move selection behavior revealed that the model did not play *in the same manner* as a human player. The breakdown of the model behavior, shown in Figure 3, appears to indicate that the model behaved more closely to that of the MiniMax algorithm than our human data. However, this similarity is likely only coincidental, as the first move selection does not align with human or MiniMax game behavior. In fact, humans and MiniMax both selected column 4 most frequently, with MiniMax selecting column 4 100% of the time and humans selecting it 67% of the time. Additionally, column 7 was the least frequently selected first move of the game for humans.

ML Model 1 Moves		Previous Move							
		-1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Current move	**1**	0	0.092415	0.114413	0.081865	0.078339	0.086855	0.078297	0.092841
	2	0	0.095361	0.111813	0.092878	0.109283	0.108957	0.09122	0.131245
	3	0	0.105302	0.063893	0.072687	0.061105	0.05855	0.079057	0.065623
	4	0	0.200295	0.193908	0.289648	0.258519	0.198914	0.312429	0.180835
	5	0	0.26215	0.174591	0.181718	0.217	0.222179	0.170277	0.229679
	6	0	0.112297	0.223626	0.153818	0.142186	0.148895	0.148613	0.139075
	7	1	0.13218	0.117756	0.127386	0.133568	0.175649	0.120106	0.160701
Human Moves		**Previous Move**							
		-1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Current Move	**1**	0.109375	0.422770	0.168014	0.064162	0.047219	0.090141	0.106803	0.143939
	2	0.059896	0.147933	0.338724	0.065896	0.078174	0.110798	0.065106	0.089015
	3	0.057292	0.093546	0.110512	0.347977	0.271773	0.083568	0.147769	0.061553
	4	0.671875	0.063089	0.133872	0.281503	0.372508	0.100469	0.050475	0.074811
	5	0.031250	0.082669	0.111411	0.076301	0.076600	0.348357	0.133138	0.125000
	6	0.044271	0.085569	0.073675	0.098266	0.121721	0.151174	0.356986	0.140152
	7	0.026042	0.104423	0.063792	0.065896	0.032004	0.115493	0.139722	0.365530

Fig. 3: Summarized data showing a comparison between the behavior of our first trained ML model (above) and human move behavior (below)

Further, when looking at the breakdown of games won and lost, this model's performance against the MiniMax algorithm was very poor, winning less than 1% of games against MiniMax at a depth of 1 and 0% at any greater search depth.

After reviewing this data, we decided to compare against additional relevant research (Kim, 2019 & Dezfouli, 2020) and found that we likely did not have enough game data to effectively model the behavior of our human players. As such, we opted to revisit our ML implementation and explore two other training approaches using all of our data combined.

SECOND ML IMPLEMENTATION

Training the second implementation for behavioral cloning was quite different. A different objective function was chosen: to predict the next move that would be taken given a board state. Additionally, all gathered data was merged into a single pool, meaning the behavior to be cloned was that of the *average player* from all conferences where data was gathered. This was done in an attempt to alleviate the issues with a lack of data that we suspected with our first approach and to determine if we could clone the *general behavior of all humans* who had played our game.

Our initial implementation of a network for this new objective was a residual network made up of convolutional layers as described in [citation for "Applying Machine Learning to Connect Four"] which was itself based on a res-net from Google's DeepMind. The structure of this network was an input convolutional layer treating the board state as a 2-channel image with a resolution of 6 x 7. We then enter the residual portion of the network where a single "block" of the network is made up of a convolutional layer, batch normalization, rectified linear unit, convolutional layer, and batch normalization - in that order. There are three of these blocks in total which then have rectified linear activation layers between them. All convolutional layers have 128 output channels, a kernel size of 3, and are padded to have identical input and output sizes to allow for the residual learning. The output block of the network consisted of a convolutional layer, batch normalization, and rectified linear activation before being flattened and passed through the linear output layer with 42 inputs and 7 outputs. The output convolutional layer was configured with 128 input channels, 1 output channel, and a filter size of 1.

Our training used cross entropy loss and the AdamW optimizer, an improved version of the Adam optimizer that uses weight decay to lower the chance of overfitting. Training was done for a fixed number of epochs that were manually tuned based on loss trends. While training the first implementation it was again determined that there was insufficient data for behavioral cloning based on the human data. Consequently, the decision was made to generate data based on our MiniMax algorithm to attempt to clone that behavior. This data would serve as a stand-in for

expert player data. This approach ensured that the amount of data available would no longer be an issue. Additionally, knowing the behavior of the MiniMax algorithm, we would also have a goal behavior to compare our results against. To that end, approximately 30,000 games were played with the AI against a random choice opponent, generating more than 330,000 board states for training.
After training, this model was able to predict the correct move ~33% of the time from 7 classes. This performance was better than the performance we observed with our previous models, but it was still poor for our purposes.

Our second revision and our current implementation of behavior cloning is based on eXtreme Gradient Boosting (XGBoost), a decision tree algorithm using an ensemble of weak learner trees that together make up a strong learner. Gradient boosting is a machine learning technique that is commonly used for regression and classification tasks (Chen, 2017). Input data for this model consisted of a single flattened board state (42 features). Major configuration variables for this approach will be listed in the appendix, values for these were found via a Tree-Structured Parzen Estimator search approach. Using the found values this network was able to produce the correct prediction with 56.72% accuracy which we found quite impressive based on 7 possible classes.

SECOND ML IMPLEMENTATION RESULTS

Our second implementation proved much more effective. The model achieved a win rate of 99.5% when playing against a random opponent in 8,000 games. The trend in move selection mirrors that of the MiniMax Algorithm. Also notable is that the model appears to have amplified the tendency to select columns on the left-hand side of the board. We also see the inverse of this with expected behaviors, highlighted in green, where the tendency to select these moves was increased by an average of 8.747%.
Additionally, when playing against the MiniMax algorithm it was able to achieve a 100% win rate up to a depth of 3. However, at a depth of 4, the win rate fell below 1% and to 0% at any greater depth. While a depth of 3 is not considered a highly difficult opponent, it is still a challenging opponent. Our previous models were only able to achieve a win rate of less than 1% against the MiniMax opponent at a depth of 1. Thus, the win rate achieved by this model and the behaviors noted in its choice of moves implies that it was, in fact, able to learn to play in a similar manner to the *expert* data it was trained on.

ML Model 2 Moves		**Previous Move**							
		-1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Current Move	**1**	0	0.049066	0.054817	0.035745	0.054209	0.046027	0.043478	0.036786
	2	0	0.04429	0.14701	0.108544	0.138387	0.031264	0.155762	0.17463
	3	0	0.148936	0.142442	0.511334	0.251653	0.111159	0.173491	0.435095
	4	1	0.449414	0.435216	0.235833	0.275892	0.621798	0.466864	0.20296
	5	0	0.215805	0.125	0.064952	0.208903	0.135475	0.086957	0.118816
	6	0	0.080764	0.089286	0.025283	0.059938	0.042987	0.060363	0.019873
	7	0	0.011724	0.006229	0.018309	0.011018	0.01129	0.013086	0.011839
MiniMax Moves		**Previous Move**							
		-1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Current Move	**1**	0	0.076346	0.071735	0.092810	0.114185	0.091935	0.067501	0.043564
	2	0	0.076346	0.154639	0.122601	0.137534	0.110798	0.171475	0.166997
	3	0	0.153374	0.134021	0.285019	0.169039	0.124948	0.169298	0.395380
	4	1	0.385140	0.308419	0.181323	0.179594	0.387380	0.354927	0.191419
	5	0	0.180641	0.144759	0.112289	0.157205	0.139156	0.097441	0.130033
	6	0	0.093388	0.109966	0.102836	0.131137	0.089010	0.090365	0.033663
	7	0	0.034765	0.076460	0.103122	0.111307	0.078980	0.048993	0.038944

Fig. 4: Summarized data showing a comparison between the behavior of our second trained ML model (above) and MiniMax move behavior (below)

CONCLUSIONS

While our first implementation was not able to effectively clone the behavior of human players, we determined this was likely due to a lack of a sufficiently large data set. However, our results do appear to show that these ML architectures are indeed capable of *learning* behaviors from data, especially in the case of the gradient boosting implementation. By using MiniMax to generate games played by an *expert*, we were able to alleviate the lack of data issue and show that our model did *learn* to play in a manner similar to the *expert* data when a sufficiently large dataset was available for training.

While training ML model to play Connect 4 in the same manner as a person or an algorithm may seem limited in its usefulness, we believe that the approach and implementation we have described could be applied to a much broader array of problems in threat modeling with relevance across the DoD M&S spectrum. Specifically, they can be used in the areas of predicting or modeling future threat behaviors based on past observed behaviors. Our approach may be able to be used to model many types of non-linear threat behaviors – from human red teams to complex cyber threat algorithms, and from maneuvering air vehicles to organized swarms of UASs. There are many areas in the DoD and M&S domains in which large amounts of historic observational data already exists and we believe this data could be leveraged to train ML models to perform the tasks that have been observed. Once such area that could benefit from this type of boosting approach could be in the creation of NPCs for training purposes. NPCs typically operate on large decision trees to imitate human-like behaviors and while this can be effective, incorporating a larger number of behaviors increases the programming complexity and hinders run-time performance of the NPCs. By utilizing our boosted approach, NPCs could not only incorporate a larger number of behaviors, but they would also be able to select appropriate behaviors more efficiently and correctly, to replicate non-linear human or threat behavior, the goal of our research project and the focus of this paper.

FUTURE WORK

We have identified several areas that could benefit from further research. First, we would like to revisit our initial ML implementation to train it on the larger dataset that was generated by the MiniMax algorithm. This couldconfirm if a lack of data was the primary issue with this approach. Next, we would like to begin testing the boosted gradient training process on more complex problems, such as more complex games and real-world DoD M&S problems. For more complicated games, we will be attempting to train models to play strategy games such as Checkers or Chess. If this proves effective, additional training could be attempted with first person perspective games.
Further, we would like to capture human motion data in VR and attempt to train a model to move and digitally-physically behave as a person does in 3-dimensional space, as this could be applied to the creation of more realistic NPC opponents for training purposes.

For real-world tasks, we will be exploring the applicability of this implementation in cyber spaces, such as cyber security threat detection, with a focus on insider threat detection. Many organizations understand the importance of securing their networks from outside threats, but detecting threats from within can be much more difficult. By applying our boosting gradient approach, we believe we can model normal human behavior on a day-to-day basis. This could be used to detect when behaviors outside of the norm are observed, giving organizations a head start to investigate a potential issue or take action before a threat becomes a problem. Finally, we believe the performance on real-world tasks, like cyber security, could be further improved with the integration of a continuous learning model, in which our model could be re-trained daily as new data is captured.

REFERENCES

Dezfouli, A., Nock, R., & Dayan, P. (2020). Adversarial Vulnerabilities of Human Decision Making. Proceedings of the National Academy of Sciences. https://www.pnas.org/doi/10.1073/pnas.2016921117

Fawkes, A. J. & Menzel, M. (2018). The Future Role of Artificial Intelligence. 27th Journal of the JAPCC. https://www.japcc.org/articles/the-future-role-of-artificial-intelligence/

Thurau, C., Sagerer, G., & Bauckhage, C. (2004). Imitation Learning at all Levels of Game-AI. Proc. Int. Conf. on Computer Games, Artificial Intelligence, Design and Education. https://www.researchgate.net/profile/Christian-Thurau/publication/228474437_Imitation_learning_at_all_levels_of_game-AI/links/09e4151290bcc65898000000/Imitation-learning-at-all-levels-of-game-AI.pdf

Andhik A. Y., Darlis H., Siti R., & Imam K. (2019). English Education Game using Non-Player Character Based on Natural Language Processing. Procedia Computer Science, Volume 161, 2019, Pages 502-508, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2019.11.158.

Warpefelt, H. (2016). The Non-Player Character: Exploring the Believability of NPC Presentation and Behavior. Department of Computer and Systems Sciences. Stockholm University. p. 118, ISSN 1101-8526

Copeland, B. J. (2020). The Modern History of Computing. The Stanford Encyclopedia of Philosophy (Winter 2020 Edition). Edward N. Zalta (ed.). https://plato.stanford.edu/archives/win2020/entries/computing-history

Chen, T. & He, T.. (2017). Xgboost: eXtreme Gradient Boosting. Microsoft (Package Version: 0.6-4). xgbhttps://cran.microsoft.com/snapshot/2017-12-11/web/packages/xgboost/vignettes/xgboost.pdfoost.pdf

Harper, J. (2015). Live, Virtual, Constructive Training Poised for Growth. National Defense Magazine.

Hartman, F. & Allen, G. (2018). Introduction to Serious Games to Enhance Defense Capabilities – Modeling and Simulation Special Edition. Cybersecurity and Information Systems Information Analysis Center. Winter 2018: Volume 5 Issue 4. https://csiac.org/articles/introduction-to-serious-games-to-enhance-defense-capabilities-modeling-simulation-special-edition/

Pollack, B. B. (2012). Creating a flexible LVC Architecture for Mixed Reality Training of the Dismounted Warfighter. Iowa State University. https://dr.lib.iastate.edu/server/api/core/bitstreams/2c413966-7af7-4b40-9c6c-d3208f8428c1/content

Kim, L., Godrej, H. & Nguyen, C. T.. (2019). Applying Machine Learning to Connect Four. http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26646701.pdf

APPENDIX

Gradient-Boosting-Configuration-Variables.svg. Available upon request (chad.etheredge@irtc-hq.com)