2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Machine Learning Techniques for Autonomous Agents in Military Simulations - Multum in Parvo

Jan Joris Roessingh,
Armon Toubman,
Joost van Oijen,
Gerald Poppinga
NLR
Amsterdam, Netherlands

Rikke Amilde Løvlid
FFI
Kjeller, Norway

Ming Hou
DRDC
Toronto, Canada

Linus Luotsinen
FOI
Stockholm, Sweden

***Abstract*—In military simulations, software agents are used to represent individuals, weapon platforms or aggregates thereof. Modeling the behavioral capabilities and limitations of such agents may be time-consuming, requiring extensive interaction with subject matter experts and complicated scripts, but nevertheless resulting in rigid, predictable performance. Autonomous agents that learn desired behaviors themselves using Machine Learning (ML) techniques can overcome these shortcomings. However, such techniques are not yet widely used and perhaps underappreciated. In this context, the latin expression "multum in parvo" ("much in little") denotes that ML agents are able to yield a large variety of behavior, despite their compactness in terms of code and usage of physical memory. This paper attempts to provide some background on applicable Machine Learning solutions and their potential military application. The paper is based on the work of the NATO Research Task Group IST-121 Machine Learning Techniques for Autonomous Computer Generated Entities.**

I. Introduction

Software agents are used in a variety of military simulation-based applications. These agents are similar to non-player characters (NPCs) in computer games and are able to autonomously sense, reason and act in a virtual environment. They are used in military training to reduce role-playing, and also in decision support applications such as planning and acquisition.

End users such as scenario developers, operators or exercise leaders at military training facilities often complain about the autonomous agents acting unrealistically or that the agent's decision making skills are too primitive and cannot be used effectively [1]. As a result, instead of using autonomous agents and to guarantee that the learning objectives are met, they rely heavily on human role-players to do the most important or critical reasoning. This approach may work well in small exercises, but it is a limiting factor in large scale exercises where many role-players are needed to stimulate the trainees, or when the task requires role-players with complex, specialized competence like fighter pilots emulating opposing forces.

Machine learning algorithms are the key enabler in a wide variety of applications today. For instance, Q&A-systems, language translation engines, search engines and recommendation systems all rely on machine learning to identify patterns in data that could not be identified using any other method. Given this background, the objective of this paper is to provide insight into, and suggest solutions for, the use of machine learning techniques to improve behavior modeling in software agents in military simulations and to facilitate end users. The insights and recommendations we provide primarily originate from our own research endeavors and experiences, but are also based on our earlier work in the NATO Research Task Group [2], [3]. For instance, in this paper we provide experimental results that indicate that machine learning techniques can be used to imitate the behavior of human role-players, learn collaborative behaviors, generate behaviors that are both creative and innovative, generate elements of surprise by making the agents learn on the fly, and learn from not only pre-processed data but also from raw video and pixels.

In Section II we provide and overview of different types of machine learning. Section III provides illustrative examples, from the NATO group's members, of solutions for behavorial modeling of software agents using these techniques. Our experiences are discussed in Section IV.

II. Machine Learning Techniques

In general, machine learning techniques can be divided in three ways. The first division is between supervised, unsupervised and reinforcement learning methods [4], [5] (Section II-A). The second division is between techniques capable of online and/or offline learning (Section II-B). The third division is between the amount of preprocessing of the data required to uncover features (Section II-C).

A. Supervised, unsupervised and reinforcement learning

Supervised methods are fed pre-labeled data, e.g. situations labeled with "correct" actions. The machine learning algorithm tries to discover the correct mapping between the situations and the labels, so that unseen situations can be acted upon. Unsupervised learning algorithms are fed unlabeled data and are for example used to find hidden patterns in data. Clustering methods, for example, excel at dividing data into categories that may not be obvious to human experts.

Reinforcement learning methods are based on "learning by doing". These methods do not learn from existing examples of "correct" actions as in supervised learning, but create their own examples while interacting with the environment. Actions that cause a change in the environment may produce a reward which indicate whether taking a certain action was good (or bad) in a certain state of the environment [6]. This reward is usually calculated by some predefined evaluation function.

In Section III we provide examples of applying both supervised and reinforcement learning methods, including evolutionary algorithms, to generate agent behavior. Unsupervised learning is not covered explicitly since it is often not suited for learning behavior models in military simulations. The specific behavior to learn is generally known beforehand and can be learned in a supervised manner using expert knowledge (as labeled data or rewards). Without any learning supervision this can easily lead to unwanted behavior models.

B. *On- and offline learning*

The second division of machine learning techniques is between techniques that can be used in an online and/or offline fashion [7]. In the context of machine learning for software agents in simulations, agents learn to map observed situations to particular actions. Online learning means that observation-action mapping is learned during operation. In turn, offline learning means this mapping is learned before the agents are deployed for their actual purpose.

Both online and offline learning have their advantages and disadvantages. Online learning methods have time constraints since learning is done during the operation of a system. However, they are also capable of learning from new, unseen situations as they arise. Though the danger of online learning is that newly learned behavior cannot be validated prior to deployment. Offline learning methods have no time constraints, as they do the learning before operation. This also allows testing the properties of the learned behavior models beforehand. Some methods are flexible and can be used both offline and online, providing the best of both worlds.

C. *Features of data*

In the applications we are discussing, agents learn to map observed situations to particular actions. An important issue is how to recognize the situation itself. Traditional methods require knowledge of features that identifies the situation and these features need to be extracted from the observed data. Deep learning is a family of machine learning methods based on learning representations of data. An observation, such as an image or video frame, can be represented in many ways such as an array of pixels, a set of edges, or shapes. Some of these representations may be easier to learn than others by traditional methods. Deep learning is a relatively new area in machine learning that refers to the use of deep neural networks that can learn directly (e.g. face recognition) from raw data examples/representations. Deep learning may be able to replace handcrafted features with efficient algorithms [8]. In Section III-C we include examples of our experiences with deep reinforcement learning (DRL) for software agents, a variant of deep learning that incorporates a reinforcement learning component.

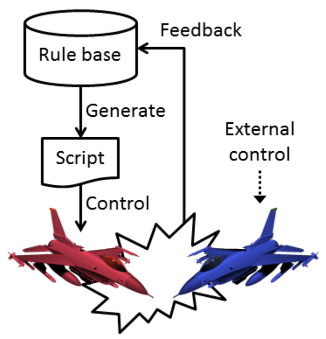

Fig. 1. Schematic view of the learning process with the DS technique. A script is generated with rules from rulebase. The script is used to control an agent. Success or failure of the controlled agent is fed back to the rules that were used.

III. Case Studies

In this section we look at different examples of machine learning applications for agent behavior. We discuss the specific techniques that are used, and how they are applied to address the needs of end users.

A. *Dynamic Scripting*

Dynamic Scripting (DS) is a reinforcement learning technique [9] which was developed to automatically generate behavior for non-player characters (NPCs) in video games. At NLR, we are investigating the use of DS for generating air combat behavior models for military training simulations.

DS works in with a rule base of predefined *if-then* rules that map situations to actions. Each rule is assigned a weight value. DS attempts to find the optimal combination of rules by which an agent can be controlled. It does so by repeatedly drawing rules from the rule base, based on their weight value. If the agent, controlled by the drawn rules, is successful in some way (i.e., it has completed some task), the rules that led to the success receive a higher weight. This increases the probability that those rules will be drawn in follow-on attempts. The learning process is shown in Figure 1.

In the context of military simulations, software agents often exist in a simulated environment that they can observe. In essence, these observations are the data that the agents can learn from, using DS. The observations typically consist of, e.g., the coordinates of the agent, the physical state of the agent (e.g., moving, airborne, damaged), and similar properties of agents that have been detected with the agent's sensors. The observations of an agent are used to check whether the if-then rules from the rulebase should fire. To keep the rules maintainable by humans, it is advisable to create higher level features from the observations. For example, an agent's position and speed vector may, when combined, indicate that an agent is about to attack. In this case, the creation of a new *is-about-to-attack* feature will allow subject matter experts to more easily write rules, compared to letting them write complicated *if*-statements checking many low-level observations.

DS can be applied both offline and online. In this context, an offline setting entails letting software agents learn to defeat other software agents (rather than human opponents). A benefit of offline learning is that many simulations can be run in a faster than real-time manner. However, not all situations that an agent might encounter can be prepared for with offline learning. In an online learning setting agents would be able to dynamically adapt their behavior to that of their human opponents, who might try to outsmart the agents in new, unforeseen ways.

At NLR we have successfully used DS to generate air combat behavior in an offline fashion. Using DS, agents are able to quickly find tactics (in the form of combinations of rules) with which other agents (using scripted behavior), could be defeated [10]. The next step is bringing these agents to human-in-the-loop simulations. In the future, DS might be applied online in these simulations, such that the agents can directly learn to counter the tactics of human participants. This way, machine learning can create challenging personalized simulations, based on the human's actions.

B. Data driven behavior modeling

Traditionally, behavior models for autonomous agents in military simulations are generated manually by interviewing subject matter experts and translating the information into code. This is time consuming and expensive. FFI and FOI are cooperating on investigating the use of machine learning to generate behavior models from examples, i.e. using offline, supervised learning. This is called data driven behavior modeling (DDBM).

Training data can be generated by recording behavior of virtual characters in example situations within an existing simulation environment. The behavior can be scripted or controlled by humans in first person shooter modus. Relevant features must be extracted from the recorded data and machine learning is used to generate behavior models that can be reused in new, similar situations.

FOI used machine learning to create autonomous agents that learn *bounding overwatch* for dismounted infantry, which is a military tactical movement used to improve the security of units when they are moving towards a target. In bounding overwatch, two units have different roles; while the bounding unit moves towards the target, it is overwatched (protected) by the halted overwatch unit. There are two types of bounding overwatch: (i) alternating bounding, and (2) successive bounds, which differ in whether the two roles are swapped or not. While in the alternate bounding the roles are swapped when the bounding unit has reached the overwatch unit's protection range, in successive bounds, the roles of bounding and overwatch unit remain the same throughout the movement.

In the FOI project, the DDBM approach is used to create autonomous agents for VBS3 (Virtual Battle Space), a game-based military simulation system [11]. The agents learn the behaviors from earlier recorded and labeled data. The labeled data is generated using scripted units in VBS3. The raw data (together with labels) are preprocessed and a dataset consisting of relevant features and corresponding labels are produced, which are used to train the behavior models by applying standard supervised ML algorithms (ID3 for decision trees and back-propagation for neural networks). The results of the study show that the agents are able to learn both bounding overwatch behaviors correctly and generalize the learned method to new unseen paths. The results from this example are presented in [12]. The main contribution of the work is that it successfully applies the DDBM concept in a real-world simulation environment as VBS3.

FFI has started experimenting with learning behavior models from data that are commonly available in different military simulation systems. The idea is that when generating behavior models based on common data, these models can be used in different simulation systems. One commonly supported standard for data models in distributed simulations is the High level architecture (HLA) [13]. Using HLA requires a Federation Object Model (FOM) that describes the data that is exchanged between the simulations systems, called federates, during execution. It is common to use standardized FOMs, known as reference FOMs, and then extend them to meet the requirements of a particular project or program. One such reference FOM that is often used in military simulation systems is the Real-Time Platform Reference FOM (RPR FOM) [14], [15].

At FFI we plan to use the RPR FOM to generate training data from recordings from human-like entities in VBS3 being manually controlled by humans and try to apply those models to entities in another simulation system called Virtual Reality Forces (VR-Forces) [16]. A federate independent of the simulation system is used to apply the learned models and command the entities in the simulation system. For controlling the entities we use low level battle management language (LLBML) as a FOM module extension to the RPR FOM. Previous publications explain the use of LLBML to control entities in military simulation systems [17], [18].

C. Deep Reinforcement Learning

Reinforcement learning (RL) is a method to solve sequential decision making problems, where an RL agent interacts with an environment over time. At each time step, the agent selects an action depending on its state and following a policy (a function that maps each state to an action). By performing the action, the agent moves to the next state and receives a scalar reward. As opposed to supervised learning, this reward may not be the consequence of a single action, but rather the consequence of a sequence of actions. Hence, there may be a latency between receiving a rewards and performing the actions that led to that reward.

The goal of the agent is to maximize the accumulated reward (discounted over time by a discount factor) from each state. The discounted accumulated reward (i.e. return) is a representation of the long-term objective of the agent. One common approach to find an optimal policy for any given state is to learn the Q-function (i.e. an action-value function that gives the expected return of taking a given action in a given state and following the optimal policy thereafter) [19].

Below we discuss two case studies for DRL, namely learning a military tactic, the Bounding Overwatch (introduced in sectionIII-B), and a serious game application for aviators.

1) Bounding Overwatch: In [20], we have evaluated the feasibility of three different deep reinforcement learning algorithms: (1) Deep Q-Learning (DQL), (2) Asynchronous Advantage Actor-Critic with a Feed Forward Network (A3C-FF), and (3) Asynchronous Advantage Actor-Critic with a Long Short-Term Memory (A3C-LSTM) to create autonomous agents for different scenarios, among others, the bounding overwatch tactic (discussed in sectionIII-B), using a simple 2D simulator, which simulates the agents and environment.

All the three algorithms (similar to other Q-learning) are model-free, that is, the autonomous agent learns the task directly using samples from the simulator, without having knowledge about the dynamics of the environment. Moreover, the visual inputs of the game are directly consumed by the algorithms and no feature extraction method is used to pre-process the input.

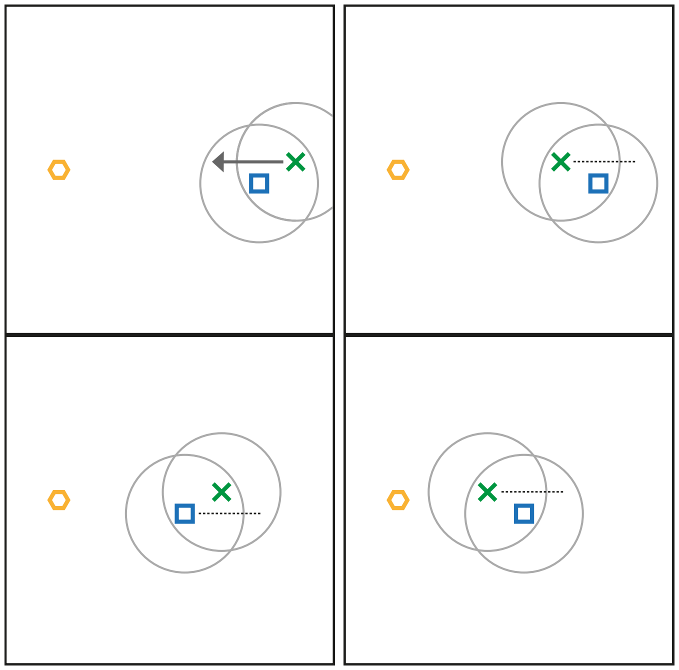
Fig. 2. From left to right and top to down; a sequence of bounding overwatch movement tactic. The orange hexagon is the target, the blue square is a programmed agent, and the green cross is the autonomous agent.

The simulator provides an interface from which the DRL algorithms can inject agent actions, retrieve a reward signal, and finally, extract image sequences representing the agents' perception of the environment. The simulator implements a basic action set consisting of five actions. Four actions that allow the agents to move in four directions within the simulated environment (east, west, north and south) and an overwatch action, which is used, when the agent halts and perform overwatch as illustrated in Figure 2. The results of the experiments for bounding overwatch (with the settings as described in [20]), shows while the DQL fails to learn the task, the A3C-FF and A3C-LSTM algorithm perform well, where the latter algorithm converges slightly faster (see Figure 3).

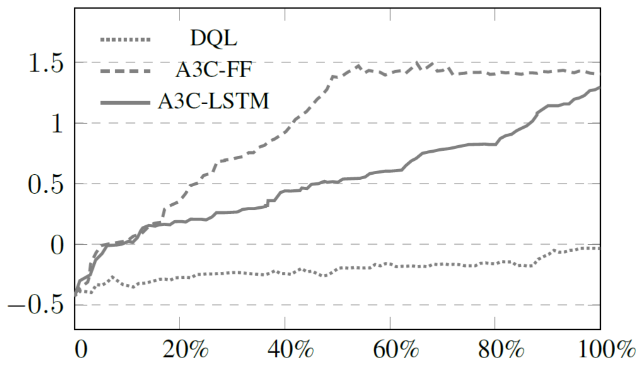

Fig. 3. Line plots representing reward over time for the bounding overwatch learning task where 0% and 100% represents the start and end of training respectively, for each learning algorithm.

2) Serious games: In this case study performed at NLR we assessed Deep Reinforcement Learning (DRL) in its ability to learn certain cognitive tasks by playing serious games. The motivation behind this study is to verify if DRL can be used to simulate a representative model for human learning for these tasks [21]. Such a model could consecutively be used for identifying training requirements, selection criteria or task design for trainees. Practical examples would be in predicting ease of human learning in case of task alterations, defining cut-off benchmarks or identifying optimal transfer-of-training for progressive part-task training. The application domain that is considered is that of Sensor Operators (SOs) for Remotely Piloted Aircraft Systems (RPAS). As a relatively new job function, new training and selection methods are required to address human factor challenges in order to improve job performance and safety [22]. A serious game in this domain focuses on (1) relevant SO tasks which include controlling sensors and detecting, identifying and tracking targets and (2) relevant cognitive abilities which include amongst others visual scanning, tracking and discrimination, spatial memory, and divided attention and vigilance to multiple sources [23].

As a candidate serious game we considered *Space Fortress* (SF) [24] which addresses some of the described abilities. SF is a well-researched serious game that was developed by psychologists for studying complex skill acquisition [24]. It has demonstrated positive transfer of training for fighter pilots[25]. SF is a challenging example for DRL. It has relatively complicated maneuvering (space dynamics); has game-rules which are difficult to infer when merely playing the game; and it includes a number of procedural tasks such as 'resource management' and 'identification friend or foe' based on a Sternberg task [26].

We assessed the performance of DRL algorithms on learning *Space Fortress*. Two DRL algorithms were tried, namely the DQN algorithm introduced in [27], and the A3C-LSTM [28]. Both algorithms did not perform well though the latter showed slightly better scorings. The poor performance can be attributed to the complicated navigation tasks in combination with sparse rewards that can be only obtained by following procedural tasks as part of the game rules.

As an alternative to learning SF, we designed a set of mini-games to represent part-task games for as SO. Three task-oriented game types were designed which are roughly illustrative for SO tasks such as navigating and orienting, aiming and shooting, and identifying and tracking targets. Each task had multiple variations to address relevant cognitive abilities, including (1) a dynamic target (requiring spatial processing abilities), (2) multiple targets (requiring divided attention), and (3) differentiating between enemies and friendlies (requiring visual discrimination). DRL was applied to these SO-mini games using the DQN algorithm. Results showed that super- or par-human performance could be achieved in all tasks for the dynamic target variation. The other variations displayed sub-human performance. To address variations that were difficult to learn, we attempted progressive part-task training (using pre-trained networks to learn new tasks). Positive transfer of training was observed, reaching up to 20% higher scores in equal total training time. Figure 4 illustrates learning curves for a task with (left) and without (right) pre-training.

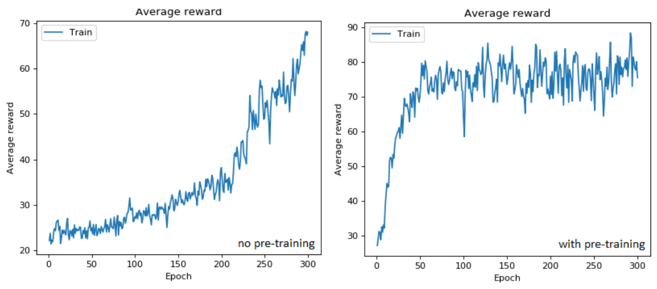

Fig. 4. Progressive part-task training comparison

Considering that DRL research on games is relatively new, it seems promising for learning tasks in relevant environments. Currently, learning complex, high-dimensional, games such as *Space Fortress* is still problematic. However, considering the pace at which improved algorithms appear in research, it can be expected that improvements will soon be made in learning the more complex, multi-dimensional serious games.

D. Evolutionary Computing

Evolutionary computing refers to a set of machine learning techniques which draw inspiration from the process of natural evolution. The four major evolutionary algorithm paradigms are evolutionary strategies, genetic programming, evolutionary programming and genetic algorithms [29].

1) Using Genetic Programming to Generate Creative Autonomous Agents: Genetic programming (GP) is an evolutionary method, inspired by biological evolution, that applies Darwin's Natural Selection Theory on a population of computer programs. In GP, each program is typically represented using a tree-structure that consists of terminal and function nodes. The initial population of the programs is randomly generated using a set of valid functions. While variation of the population is maintained by simulated reproduction and mutation, in each generation those individual programs that show a higher performance against a designed fitness criterion will survive [30], [31]. In [32], we provide an example of using GP for generating autonomous agents that show a creative behavior. The work provides experimental results using a predator/prey game and show that predator behavior created by GP surpasses predator behavior manually programmed by humans and argues that the automatically generated agent is unlikely to be generated by a human.

2) Learning Classifier Systems: Learning Classifier Systems (LCSs) are a family of genetic-based machine learning methods [33]. In general, an LCS attempts to optimize classification or action selection through evolution of rules in a rulebase. As they are evolutionary systems, one of the strengths of LCSs is their ability to come up with creative solutions to problems. Therefore, an agent using an LCS to learn to defeat other agents should be able to come up with clever new tactics that counter the tactics of its opponents. Furthermore, because of their rule-based foundation, LCSs can produce tactics that should be readable by human experts.

In comparison to dynamic scripting (DS) (introduced in section III-A), both LCSs and DS operate on rules, yet in different ways. An LCS works by mutating rules, while a DS system works by finding optimal combinations of given rules. Because of their compatible paradigms, the NLR has investigated a combination of an LCS with DS to generate air combat tactics. We envisioned a complete solution wherein an LCS would generate many high quality behavior rules in an offline fashion. These rules would be the constituent parts of tactics that are discovered to be effective against a set of opposing agents. These rules could then be put into a rulebase and given to DS. DS would then be able to form new tactics by recombining rules, thereby providing challenging opposition to e.g. human trainees in an air combat simulator.

IV. Discussion and Conclusion

In this paper we have presented various cases in which machine learning (ML) was used to generate behavior for autonomous agents in military simulation systems.

- Dynamic Scripting, a relatively simple and transparent rule-based reinforcement learning technique, was applied to Air-to-Air combat. This method requires domain expertise to start with (i.e. is not model-free), but knowledge can straightforwardly be added in the form of if-then rules. It is suitable for off- and online learning, the latter form enabling personalized simulation.
- Data Driven Behavior Modeling is a supervised learning technique that was applied to tactical maneuvering of soldiers. It generalizes well to unseen examples. Moreover, example data for DDBM collected in one simulation package (e.g., VBS) could be used to train entities in a second simulation package (e.g., VR-Forces), using HLA.
- Deep Reinforcement Learning is a neural network-based, model free, reinforcement learning technique, and has been applied to tactical maneuvering of soldiers. It has also been applied to model the learning process and

task performance of RPAS crew in various settings. It works well on raw data. More advanced versions than the baseline version (DQN) are needed from more complex high-dimensional environments. Unfortunately, neural networks are not easy readable, hence remain somewhat of a black box to the end user.

- Different evolutionary computing algorithms have been investigated. Genetic programming has been applied to learning a predator the tactical task of devouring a flock of prey. Eventually, the predator is able to implement a superior tactic that will not be easily implemented by human experts. Another form of an evolutionary algorithm is called a Learning Classifier System and has been applied to a air-to-air combat. This rule-based evolutionary technique allows creation of new rules using mutation and cross-over, while evaluating their fitness in combat situations. The clever behavioral solutions these techniques come up with, are generally readable, but may not always seem logical to the end user.

This paper supports the hypothesis that incorporating ML in autonomous software agents allows for richer behaviors in complex environments. Such agents are better tailored to the knowledge and skills of the trainee. Moreover, these ML agents enable automatization of the scenario development process and are relatively compact in terms of code and usage of physical memory, therewith facilitating the end user.

References

[1] N. Abdellaoui, A. Taylor, and G. Parkinson, "Comparative analysis of computer generated forces' artificial intelligence," DEFENCE RESEARCH AND DEVELOPMENT CANADA OTTAWA (ONTARIO), Tech. Rep., 2009.

[2] A. Toubman, G. Poppinga, J. J. Roessingh, M. Hou, L. Luotsinen, R. A. Løvlid, C. Meyer, R. Rijken, and M. Turčaník, "Modeling cgf behavior with machine learning techniques: Requirements and future directions," in *Proceedings of the 2015 Interservice/Industry Training, Simulation, and Education Conference*, 2015, pp. 2637–2647.

[3] A. Toubman, J. J. Roessingh, J. van Oijen, R. A. Løvlid, M. Hou, C. Meyer, L. Luotsinen, R. Rijken, J. Harris, and M. Turcanık, "Modeling behavior of computer generated forces with machine learning techniques, the nato task group approach," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2016, 2016.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge Univ Press, 1998, vol. 1, no. 1.

[7] R. S. Sutton, S. D. Whitehead *et al.*, "Online learning with random representations," in *Proceedings of the Tenth International Conference on Machine Learning*, 1993, pp. 314–321.

[8] H. A. Song and S.-Y. Lee, "Hierarchical representation using nmf," in *International Conference on Neural Information Processing*. Springer, 2013, pp. 466–473.

[9] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.

[10] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and H. J. van den Herik, "Rapid adaptation of air combat behaviour," in *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, ser. Frontiers in Artificial Intelligence and Applications, G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, Eds., vol. 285. The Hague, The Netherlands: IOS Press, 29 August-2 September 2016 2016, pp. 1791–1796.

[11] Bohemia Interactive. (2016) VBS. [Online]. Available: https://bisimulations.com/virtual-battlespace-3

[12] F. Kamrani, L. Luotsinen, and R. A. Løvlid, "Learning objective agent behavior using a data-driven modeling approach," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2016.

[13] NATO NSA, *STANAG 4603 - Modelling and Simulation Architecture Standards for Technical Interoperability: High Level Architecture (HLA)*, 2nd ed., 2015.

[14] Simulation Interoperability Standards Organization (SISO), *Standard for Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0*, http://www.sisostds.org, 2015, SISO-STD-001.1-2015.

[15] ——, *Standard for Guidance, Rationale, and Interoperability Modalities (GRIM) for the Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0*, http://www.sisostds.org, 2015, SISO-STD-001-2015.

[16] VT MAK. VR-Forces. [Online]. Available: https://www.mak.com/products/simulate/vr-forces

[17] A. Alstad, O. Mevassvik, M. Nielsen, R. Løvlid, H. Henderson, R. Jansen, and N. de Reus, "Low-level battle management language," in *Proceedings of the 2013 Spring Simulation Interoperability Workshop*, no. 13S-SIW-032, 2013.

[18] J. Ruiz, D. Dsert, A. Hubervic, P. Guillou, R. Jansen, N. de Reus, H. Henderson, K. Fauske, and L. Olsson, "BML and MSDL for multi-level simulations," in *Proceedings of the 2013 Fall Simulation Interoperability Workshop*, no. 13F-SIW-002, 2013.

[19] Y. Li, "Deep reinforcement learning: An overview," *CoRR*, vol. abs/1701.07274, 2017. [Online]. Available: http://arxiv.org/abs/1701.07274

[20] B. Toghiani-Rizi, F. Kamrani, L. J. Luotsinen, and L. Gisslén, "Evaluating deep reinforcement learning for computer generated forces in ground combat simulation," in *Submitted to IEEE International Conference on Systems, Man, and Cybernetics*, 2017.

[21] J. van Oijen, G. Poppinga, O. Brouwer, A. Aliko, and J. J. Roessingh, "Towards modeling the learning process of aviators using deep reinforcement learning," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2017, 2017.

[22] W. R. Howse, "Knowledge, skills, abilities, and other characteristics for remotely piloted aircraft pilots and operators," DTIC Document, Tech. Rep., 2011.

[23] W. L. Chappelle, K. McDonald, and R. E. King, "Psychological attributes critical to the performance of mq-1 predator and mq-9 reaper us air force sensor operators," DTIC Document, Tech. Rep., 2010.

[24] A. Mané and E. Donchin, "The space fortress game," *Acta psychologica*, vol. 71, no. 1-3, pp. 17–22, 1989.

[25] D. Gopher, M. Well, and T. Bareket, "Transfer of skill from a computer game trainer to flight," *Human Factors*, vol. 36, no. 3, pp. 387–405, 1994.

[26] S. Sternberg *et al.*, "High-speed scanning in human memory," *Science*, vol. 153, no. 3736, pp. 652–654, 1966.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 02 2015.

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016.

[29] A. E. Eiben, J. E. Smith *et al.*, *Introduction to evolutionary computing*. Springer, 2003, vol. 53.

[30] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.

[31] ——, *Genetic programming II: automatic discovery of reusable programs*. Cambridge, MA, USA: MIT Press, 1994.

[32] L. J. Luotsinen, F. Kamrani, P. Hammar, M. Jändel, and R. A. Løvlid, "Evolved creative intelligence for computer generated forces," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2016.

[33] J. H. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Goldberg, S. Forrest, R. L. Riolo, R. E. Smith, P. L. Lanzi, W. Stolzmann *et al.*, "What is a learning classifier system?" in *Learning Classifier Systems*. Springer, 2000, pp. 3–32.