Developing Combat Behavior through Reinforcement Learning in Wargames and Simulations

Jonathan Boron
MAGTF Staff Training Program
United States Marine Corps
Quantico, USA
jboron90@gmail.com

Chris Darken
Computer Science
Naval Postgraduate School
Monterey, USA
cjdarken@nps.edu

***Abstract*—Progress in artificial intelligence (AI), particularly deep reinforcement learning (RL), has produced systems capable of performing at or above a professional-human level. This research explored the ability of RL to train AI agents to achieve optimal offensive behavior in small tactical engagements. Agents were trained in a simple, aggregate-level military constructive simulation with behaviors validated with the tactical principles of mass and economy of force. Results showed the combat model and RL algorithm applied had the largest impact on training performance. Additionally, specific training hyper-parameters also contributed to the quality and type of observed behaviors. Future work will seek to validate RL performance in larger and more complex combat scenarios.**

Keywords—wargames, constructive simulations, artificial intelligence, reinforcement learning, agent combat behavior

I. Introduction

Deep RL has steadily progressed in capability over the past several decades, as trained systems have defeated world-champion human players in the games of chess, Go, and poker [1]. Moreover, DeepMind and OpenAI have achieved noteworthy milestones in real-time strategy (RTS) and related games, producing systems trained in StarCraft II and Dota 2, respectively, that have outperformed over 99% of human players [2], [3]. These latter accomplishments are of particular interest to the military domain, as RTS games share many characteristics with military wargames and constructive simulations, such as large action spaces, fog of war, and long-term planning. However, unlike military constructive simulations, RTS games abstract many important aspects of combat, to include logistical and intelligence functions. RTS games also generally have a large community of players against which performance can be compared, whereas, only a small cadre of individuals are expertly adept in any one military simulation. Consequently, this research sought to validate the performance of RL in small tactical engagements in a Python-coded environment resembling a constructive military simulation.

As engagements increase in size and complexity, it becomes more difficult to evaluate RL behavior and assess whether it has discovered the global optimum for performance. For this reason, small tactical scenarios were chosen, with the assumption that units had complete command and control, logistics, and battlefield intelligence. These assumptions were made so that the optimal solutions to these scenarios were apparent through inspection or simple derivation. It should be noted that optimal behavior and tactical behavior are two distinct concepts. In this research, engagements were designed such that optimal solutions aligned with tactical behavior and corresponded to one of two principles of war – mass and economy of force. Per Marine Corps Doctrinal Publication (MCDP) 1-0 *Operations* [4], mass is defined as "[T]he concentration of friendly capabilities at the decisive place and time to achieve decisive results." It is also commonly referred to as consolidation of force. MCDP 1-0 [4] describes economy of force as allocating the minimum essential combat power to secondary efforts in order to achieve superiority at the primary objective.

II. Methodology

A. Training Environment

The environment used to train agents consisted of a 10-by-10 featureless plain divided into hexagonal tiles. Similar to how movement is executed in many analog wargames, agents were capable of moving in one of six directions, represented by the 60-degree increments of a compass. Agents did not need to take any explicit action to begin combat, as combat would start once two or more opposing entities were adjacent to each other. Upon initiation of combat, entities were locked in place and unable to move until that particular engagement was complete. In total, scenarios would proceed until one side was destroyed, the scenario timed out, or an agent-controlled entity retreated outside of the map boundaries.

The simulation was executed in a time-stepped, turn-based manner. RL agents controlled all attacking, offensive entities, as opposing defensive entities remained static in fixed positions. All entities represented aggregate troop formations of company- or platoon-size, the equivalent of 150 and 50 personnel, respectively. The simulation was executed in a time-stepped, turn-based manner. Each agent received a turn in sequence, and could move to any unoccupied adjacent hex if not engaged. Since defensive entities remained static, they were not provided a turn to move. For example, if the agent controlled three entities in a fight against two static entities, then it would take three turns for all three entities to move a single space. The primary impact of this movement model was in combat, since a minimum of one turn, or time-step, separated all entity engagements. However, combat and casualties were resolved at the end of each turn, considering all entities currently engaged.

B. Combat Models

Four combat models, each with their own reward system, were tested to resolve the outcomes of engagements. The first was the probability of kill model, which simply drew a uniformly distributed float over [0,1) and compared it to a pre-established value. If the random float was less than this value, then the opposing entity would be destroyed. A reward of +1.0 was provided to an agent if it destroyed an opposing entity, but a penalty of –0.6 was applied if an agent-controlled entity was killed.

The second combat model was a referee system, in which an engagement would end once two agent-controlled entities converged on a single opposing entity. In this regard, there was no explicit combat resolution, as the purpose of this combat model was to entice agents to quickly mass forces on an opponent. Consequently, agents were provided rewards based on the time it took to converge two entities on to an opponent. If the time between entity engagements was 1.0, the fastest attainable, then a reward of +1.0 was provided. Otherwise, the reward would decrease in a linear fashion until the time between engagements reached 4.0 or longer, at which time a reward of 0.0 was provided.

The last two combat models are directly related, as they are the deterministic and stochastic variations of Lanchester's modern combat equations. Unlike the combat models previously discussed, Lanchester equations consider the size and attrition rate of the two opposing forces [5]. As such, attrition rate is defined as the number of casualties a force inflicts on its opponents per capita per unit time [5]. The equation for the deterministic Lanchester combat equation is as follows:

$$\Delta B = A \times a \times \Delta t \tag{1}$$

Where ΔB was the change in opposing force size, A was the size of the attacking force, a was the attrition rate of the attacking force, and Δt was the change in time. The stochastic Lanchester combat model, shown in equation (2), was similar to its deterministic counterpart, although an additional term was added that represented a uniformly distributed random float over [0,1).

$$\Delta B = A \times a \times \Delta t \times \xi \tag{2}$$

In this research, attrition rates for all forces were set to 0.05 and the change in time was always set to 1.0. Initial force sizes were either 150 or 50, as discussed above.

The reward provided to an agent was based on several factors. The first was the ratio of remaining forces to initial forces, such that an agent received a higher reward if it managed to preserve more of its force size. The second factor was the size of the unit defeated. If the agent destroyed a company (150 personnel), it received the full reward. However, if it destroyed a platoon (50 personnel), it would receive one-third of the original reward. Equation (3) describes the reward system implemented for the Lanchester combat models.

$$R = (f / F) \times (E / 150) \tag{3}$$

Where R was the reward, f was the final friendly force size, F was the initial friendly force size, and E was the initial enemy force size. On the contrary, for both variants of the Lanchester combat model, an agent received a flat penalty of –0.05 if one of its entities were destroyed.

C. Machine Learning Tools

Two primary RL resources were used to ease the implementation of RL algorithms and conduct experiments. These resources consisted of Spinning Up and Gym, both of which are developed and maintained by OpenAI. Spinning Up is a resource for deep RL researchers that provides a suite of software tools and algorithm background information [6]. The software component of Spinning Up allows a training environment to be connected to various RL algorithms and computational libraries. On the other hand, Gym is a toolkit that allows researchers to build and compare RL algorithms [7]. It comes with a standard set of training environments, but also establishes the architecture for users to construct their own. Gym effectively provides a common interface for environments and RL algorithms to interact.

Feed-forward neural networks were used to control attacking entities. Convolutional neural networks were considered but Spinning Up was not well suited for raw visual feed. Spinning Up was capable of connecting to both TensorFlow and PyTorch, although PyTorch was the primary computational library used for this project. While a number of training and neural network hyper-parameters were examined in this research, rectified linear units were kept as the activation functions for all experiments. Additionally, three RL algorithms were tested; these include Vanilla Policy Gradient (VPG), Proximal Policy Optimization (PPO), and Trust Region Policy Optimization (TRPO).

D. State Representation

Neural networks were provided four inputs to the state space. This included the global positions of the entity currently in-action, all friendly entities, all opposing entities, and any entities engaged in combat. Neural networks were not provided pin-point locations for every entity, but rather a "fuzzy" representation that weighed each entity location with respect to a discretized 10-by-10 map grid of squares.

E. Training Scenarios and Evaluation

RL agents were trained and evaluated in three different force configurations. Fig. 1 displays an example of each configuration.

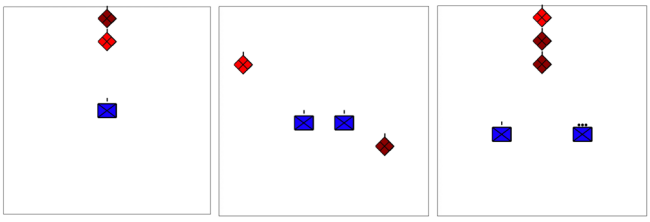

Fig. 1. Examples of Each Force Configuration Tested. Source: [8]

The first and smallest was a two-versus-one scenario, depicted in the far-left image of Fig. 1, in which two AI-controlled companies fought a single static opposing company. The second scenario, in the center of Fig. 1, was a two-versus-two configuration, with two AI-controlled companies against two static opposing companies. The last scenario was three-versus-two, presented in the far-right image of Fig. 1. In this case, three AI-controlled companies were pitted against a single opposing platoon and company.

For the first two scenarios depicted in the left and middle of Fig. 1, AI-controlled companies would begin each engagement at a random location on the map. However, for the third scenario, the AI-controlled companies always began in a column formation at the top of the map. In all scenarios, the opposing blue entities remained centered in the middle of the environment.

In order to comprehensively compare agent performance to the tactical principles of mass and economy of force, several metrics were examined. As it pertained to mass, the mean undiscounted reward and percentage of perfect teaming were the primary indicators used to assess behaviors. In this context, percent perfect teaming refers to the percentage of time in which the time between engagements was 1.0. On the other hand, for economy of force, the mean discounted reward was the primary metric used to validate performance. Discounted reward was determined to be the more appropriate measure of performance, since this tactic did not always result in the maximum raw reward.

III. Results

Two statistical methods were used to analyze results. Regression models were built to assess the impact of the various learning and neural network hyper-parameters, and two-direction two-sample t-tests were conducted to detect differences between data sets. An alpha level of 0.05 was used for all statistical tests. All means are presented with their respective standard deviations (StDev).

A. Mass

RL agent performance was validated with the principle of mass in two scenarios, the two-versus-one and the two-versus-two. The experiment conducted for the two-versus-one configuration was the largest, as a total of 528 design points were replicated five times for a total of 2640 trials. TABLE I. presents the respective factors and levels for this experiment.

TABLE I. Full Factorial Design for Two-versus-One Experiment

Factor	Levels
Learning Rate	0.01, 0.005, 0.001, 0.0005, 0.0001
Hidden Units	32, 2 × 32, 64, 2 × 64, 96, 2 × 96
Training Duration	500, 1000
RL Algorithm	PPO, VPG, TRPO[a]
Combat Model	Probability of Kill, Referee, Deterministic and Stochastic Lanchester

[a.] TRPO was only used with the Deterministic Lanchester combat model.

Results for this experiment showed that the combat model employed had the greatest effect on performance, as deterministic combat models generally performed better than stochastic ones. TABLE II. summarizes the top result for each combat model. Because different reward functions were employed for every combat model, a comparison between average rewards was not conducted. Thus, only the average perfect teaming percentages were compared.

TABLE II. Top Results for Two-versus-One Experiment

Combat Model	Highest Mean Result (± StDev) (Perfect Teaming Percentage)
Probability of Kill	72.6 ± 1.7
Referee	100.0 ± 0.0
Deterministic Lanchester	93.0 ± 2.4
Stochastic Lanchester	96.1 ± 1.4

With respect to the RL algorithm, PPO consistently outperformed VPG in stochastic environments, as was verified using two-sample t-tests to compare average reward and perfect teaming percentages. However, RL algorithm performance in deterministic combat models were similar. TABLE III. presents the top performance for each RL algorithm.

TABLE III. Comparison of VPG and PPO in Two-versus-One Experiment

Combat Model	RL Algorithm Highest Mean Result (± StDev) (Perfect Teaming Percentage)	
	VPG	PPO
Probability of Kill	65.1 ± 3.6	72.6 ± 1.7
Referee	97.1 ± 2.4	100.0 ± 0.0
Deterministic Lanchester	93.0 ± 2.4	91.5 ± 2.0
Stochastic Lanchester	87.8 ± 2.6	96.1 ± 1.4

In terms of training hyper-parameters, learning rate and training duration had the greatest influence, as the larger learning rates and longer training produced superior agents. In only one instance was the number and size of hidden layers found to have an effect.

A comparison of VPG, PPO, and TRPO was also made in the deterministic Lanchester combat model. A three-way evaluation of percent perfect teaming using two-sample t-tests showed that VPG and PPO performed similarly with a p-value of 0.912. However, TRPO was found to outperform both VPG and PPO in this environment, with p-values of 0.039 and 8.3 x 10^{-5}, respectively.

The next experiment examined agent performance in the two-versus-two scenario, this design is contained in TABLE IV.

TABLE IV. Full Factorial Design For Two-versus-Two Experiment

Factor	Levels
Learning Rate	0.01, 0.005, 0.001, 0.0005, 0.0001
Hidden Units	32, 2 × 32, 64, 2 × 64, 96, 2 × 96
Training Duration	500, 1000
Combat Model	Deterministic and Stochastic Lanchester

For the two-versus-two experiment PPO was the only RL algorithm applied. Moreover, only the Lanchester combat models were implemented. This was because the referee model was not well suited for scenarios with more than one static opponent, and the probability of kill model had previously produced inferior results in the two-versus-one experiment.

Overall, the deterministic Lanchester combat model outperformed its stochastic counterpart. The best performing configuration in the deterministic combat model had a perfect teaming percentage of 97.2% ± 0.8, whereas the best configuration in the stochastic model had a percentage of 88.5% ± 6.1. A two-sample t-test comparing the two sets of perfect teaming percentages confirmed this observation, producing a p-value of 2.23 x 10^{-5}. The learning rate was the only hyper-parameter identified to have a significant impact in both models. However, the larger learning rates performed better in the stochastic model as opposed to smaller learning rates in the deterministic.

B. *Economy of Force*

The three-versus-two configuration was tested with the following experimental design. In this case, only PPO and deterministic Lanchester were implemented.

TABLE V. FULL FACTORIAL DESIGN FOR ECONOMY OF FORCE EXPERIMENT

Factor	Levels
Learning Rate	0.01, 0.005, 0.001, 0.0005, 0.0001
Hidden Units	32, 2 × 32, 64, 2 × 64, 96, 2 × 96
Discount Factor	0.995, 0.99, 0.985, 0.98, 0.975, 0.97, 0.965

The RL agents trained in this experiment exhibited two unique behaviors – economy of force and mass. The discount factor was found to have the greatest influence on this observation. For discount factors lower than 0.98, agents generally exhibited economy of force, as they split their force into two groups. The first group, consisting of two companies, attacked the opposing company, while the second group, composed of a single company, simultaneously attacked and destroyed the opposing platoon. For discount factors higher than or equal to 0.98, mass tended to be the favored tactic, with all three AI-controlled companies sequentially attacking the opposing company and then platoon.

Analysis of the size and timing of rewards revealed why the discount factor had a causal effect on which principle of war was exhibited. An agent would receive larger rewards for applying the principle of mass, but at later intervals. Thus, as the discount factor was decreased, the rewards attained with mass, though greater, became less valuable to the agent. This effect is displayed in Fig. 2.

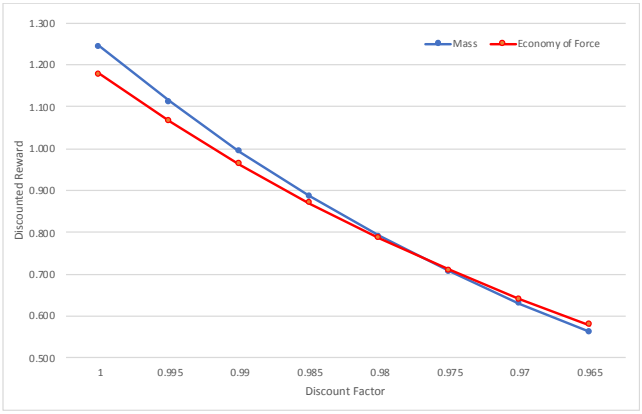

Fig. 2. Impact of Discount Factor on Reward. Source: [8]

Across all discount factors, the top performing agent for mass attained a mean undiscounted reward of 1.2415 ± 0.0002 and percent perfect teaming of 99.86% ± 0.17. The top performing agent for economy of force achieved a mean discounted reward of 0.6370 ± 0.0043, a difference of 0.5% when compared to the maximum discounted reward of 0.640.

IV. CONCLUSIONS AND FUTURE WORK

From this research, it was deduced that the type of combat model used, either deterministic or stochastic, had the most significant impact on AI agent performance. With exception to the stochastic Lanchester in the two-versus-one scenario, deterministic combat models produced higher performing agents. With respect to the RL algorithms implemented, PPO outperformed VPG in stochastic combat models, although their performance in deterministic environments were similar. Moreover, TRPO also showed promise, producing better results than PPO and VPG in the deterministic Lanchester. However, this dominance cannot be extrapolated to stochastic combat models without further experimentation. Lastly, the learning rate and training duration were the only hyper-parameters found to have a significant impact on performance. While the size and number of hidden layers occasionally generated an effect, no trends could be identified.

Future work will look to expand the size and complexity of these scenarios, as many features of combat were highly simplified or neglected. The easiest modification in this regard is to increase the number and types of units, as the entities in this research were strictly homogenous. Furthermore, providing the AI agents' opponents with the ability to react and maneuver will also create a more robust combat simulation. With respect to the environment itself, increasing the size of the map and adding various terrain and weather effects are important factors. Lastly, AI agents should be trained in a broader set of tasks and missions. This research focused on entities conducting offensive operations, but other missions, such as a defense or raid, should be considered as well.

REFERENCES

[1] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, Jan. 2018, doi: 10.1126/science.aao1733.

[2] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019, doi: 10.1038/s41586-019-1724-z.

[3] OpenAI, "OpenAI Five," Jun. 25, 2018. [Online]. Available: https://openai.com/blog/openai-five/

[4] *Operations*, Marine Corps Doctrinal Publication 1–0, Headquarters Marine Corps, Washington, DC, 2017. [Online]. Available: https://www.marines.mil/Portals/1/Publications/MCDP%201-0%20Marine%20Corps%20Operations.pdf

[5] A. Washburn, "Lanchester Systems," Defense Technical Information Center, Fort Belvoir, VA, Apr. 2000. doi: 10.21236/ADA383409.

[6] OpenAI, "Welcome to Spinning Up in deep RL! — Spinning Up documentation." Accessed Jan. 09, 2020. [Online]. Available: https://spinningup.openai.com/en/latest/

[7] OpenAI, "Gym: A toolkit for developing and comparing reinforcement learning algorithms." Accessed Jan. 09, 2020. [Online]. Available: https://gym.openai.com

[8] J. Boron, "Developing combat behavior through reinforcement learning." M.S. Thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2020.