2021

Training Intelligent Red Team Agents Via Reinforcement Deep Learning

Ballard, Marcus A.

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/69898

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

TRAINING INTELLIGENT RED TEAM AGENTS VIA

REINFORCEMENT DEEP LEARNING

by

Alan Ballard

January 2022

Prepared for: Marine Corps Systems Command (MARCORSYSCOM). This research is supported by funding from the Naval Postgraduate School, Naval Research Program (PE 0605853N/2098). NRP Project ID: NPS-21-M079-B

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Wargames are an essential tool for education, training, and formulation of strategy. They are especially important in the evaluation of threats from, and strategies against, trained adversaries who present significant risk to friendly forces. We proposed to develop a wargame adversary trained to defeat the current strategy of friendly forces, allowing the evaluation of alternate strategies against an intelligent, simulated opponent. This research sought to evaluate the ability of different deep neural network algorithms to train an enemy red team against a friendly blue team with an existing strategy, in terms of both efficacy and efficiency, and the resiliency of the trained red team to subsequent changes in blue team strategy.

A simulated combat environment was created in which a blue team was first trained using deep reinforcement learning to defeat a stationary opponent in an open battlefield, establishing a baseline blue strategy. The red team was then trained, again with deep reinforcement learning, to defeat this blue team, after which the blue team's strategy was altered, and the two teams were allowed to engage in combat again. During this experiment, the time required to train the red team and the proportion of combat outcomes that red won were calculated and later analyzed using linear models. Several important variables were identified, from both the combat environment and the algorithms employed to train each team, that significantly affected the time required to train the red team and the red team's subsequent ability to win against an opponent with an altered strategy. Additionally, it was determined that while there was no significant difference between the algorithms in the mean time required for red team training, the choice of algorithm had a significant effect on the red team's subsequent ability to win against an opponent whose strategy differs from the one against it was trained.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. ROLE AND DESIGN OF WARGAMES

“The core attribute of a good wargame is an adversarial environment…”
(Pournelle, 2017, p.51).

A wargame is simulated combat in an artificial environment in which players take actions and then observe the consequences of their actions. In that sense, wargames are an important tool for military forces to evaluate current or potential strategies and capabilities in a risk-free environment. Simulations such as wargames provide a window into the interaction of opposing systems and are useful for deriving insight into how decisions are made in the face of uncertainty (Pournelle, 2017).

These simulations have historically been conducted by assembling a group of subject matter experts to play as blue (friendly) and red (enemy) agents. Each agent takes an action – moving forces, firing on the enemy’s forces, etc. – and the consequences of that action are determined – forces are successfully moved, enemy forces lose 10% combat strength, etc. The method of determining the consequences might be agreement among players, decision of a subject matter expert, correlation, or interpolation of historical trends (Wade, 2018).

According to (Wade, 2018), “wargames are best used for unstructured problems where human decision making is central to the issue being addressed” and “a wargame is effective because it focuses on human decision making in a competitive sequential environment”. While it is true that wargaming has heretofore been a human endeavor, artificial intelligence (AI) has begun to take its own role in wargaming.

B. DEEP REINFORCEMENT LEARNING

At its simplest, reinforcement learning involves a trial-and-error approach in which an artificial agent takes an action under uncertainty given its understanding of its environment (the “state”), and receives a reward, possibly negative. The agent learns whether the action was good or bad according to the reward received and updates its understanding of the environmental state accordingly before taking another action and repeating the cycle until a pre-specified goal is achieved. The agent’s primary objective is to maximize the sum of these rewards, thus learning the "best" means of achieving its goal.

In a discrete environment, there are *m* possible states and *n* possible actions. Q-Learning (Watkins, 1989) populates a m x n "Q table" containing the maximum expected future reward for each possible combination of states and actions in the environment. At some time *t*, the environment will be in state s_t and the agent may take action a_t. If so, the expected future reward for all subsequent actions can be found using the Bellman equation, $Q(s_t, a_t) = E[R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \ldots]$, where R_t is the reward at time *t* after taking action a_t from state s_t, R_{t+1} is the reward after the subsequent step, and so on, and γ is the discount rate applied to the reward. The discount rate is an adjustable parameter that signals to the agent how to value rewards over time. Higher values of γ indicate to the agent that long-term rewards are more important when selecting actions while smaller values place more importance on short-term rewards.

The Q table is initialized with random values, the agent selects the action that currently yields the maximum expected reward from the current state, $R(s,a)$. The algorithm then updates the expected future reward for that state and action using this equation:

Updated $Q(s,a) = Q(s,a) + \alpha * [R(s,a) + \gamma * \max\{Q'(s',a')] - Q(s,a)\}]$,
where $\max[Q'(s',a')]$ is the maximum expected future reward among all other possible states and actions and α is an adjustable learning rate that controls how quickly the agent will explore the environment. This process repeats until a pre-specified goal is achieved or some stopping criteria is met.

After population, the Q table will yield a clear "policy" that tells the agent which action will yield the highest expected future reward from a given state. However, this method faces two major challenges. First, the table must contain all possible states and actions that can exist within the environment. An exhaustive list of environmental states quickly becomes prohibitive as the dimensions of the environment grows large. Second, as the number of possible states grows large, the number of state/action combination which an agent might be expected to actually encounter grows small. Thus, the Q table will be sparsely populated with rewards and therefore ineffective.

Q-Learning experienced a breakthrough when DeepMind introduced Deep Q-Network (DQN) in 2015 (Mnih et al, 2015). DQN combines Q-learning with a deep neural network (DNN) to learn a low-dimensional representation of a high-dimensional environment. In DQN, a neural network (NN) with multiple layers between the input and output layers - a “deep” neural network - is created with randomly assigned parameters **θ**. The inputs to the network are the states of the environment and the outputs of the network are probabilities assigned to each possible action from that state. When the agent encounters a given state, it will select the action with the highest associated probability. However, with randomly initialized parameters, the network will not be very accurate at first and tends to recommend actions that lead to sub-optimal rewards. This error can be quantified in a loss function:

$$\text{Loss} = [R(s,a) + \gamma * \{\mathrm{Q}'(s',a')] - \mathrm{Q}(s,a)\}]^2$$

If the reward from the action taken is close to the maximum possible reward obtainable from the optimal action, the loss will be small. If the network recommends an action that is significantly less rewarding than the optimal action’s reward, the loss will be large. Therefore, a reasonable strategy to identify optimal actions is to constrain the network to recommend actions that result in minimal loss. This is done by using a technique called “back propagation” where the gradient of the loss function ∇Loss(**θ**) is calculated with respect to each parameter in the network using the chain rule from calculus. The parameters of the networks are then updated as:

$$\text{New } \boldsymbol{\theta} = \boldsymbol{\theta} - \alpha * \nabla\text{Loss}(\boldsymbol{\theta})$$

With parameters updated, the DQN model then recommends the next action for the agent to take and repeats the process. Thus, as the agent explores the environment, the DNN incrementally improves its ability to predict which action is best in a given situation.

This approach is particularly suitable to simulations. Given a goal, an agent may be allowed to play in a simulated environment many, many times until the model develops a sense of which actions are best. In fact, DQN and later evolutions of DQN have learned to play some games well beyond human capabilities against both computer and human competitors, such as Atari (Mnih et al, 2013), Chess (Silver et al, 2018) and mass multiplayer online games Dota2 (Berner et al, 2019) and StarCraft II (Vinyals et al, 2019). According to (Wade, 2018), the four critical parts of a military wargame are the players, the scenario, the rule set, and the adjudication method. Additionally, a well-designed wargame should include uncertainty, fair competitive environment, adjudication of and consequences for actions, and iterations (Pournelle, 2017). Therefore, it is natural that DRL would be applied when developing artificially intelligent agents for use in wargames as the DRL agents themselves learn to operate in a defined environment with pre-specified goals, competitive players, actions decided under uncertainty and consequences for those actions.

Further, there is an international effort to apply AI in general, and DRL in particular, to military wargames. In Australia, (Moy and Shekh, 2019) investigated the ability of combining AlphaZero deep reinforcement learning and supervised learning to automatically learn to play wargames. In China, (Wang et al, 2020) investigated the large-scale use of DRL in wargames. (Zhang and Xue, 2020) proposed an actor-critic framework for AI decision making using Convolutional Neural Networks. In the UK, researchers proposed decision-making software that uses AI to allocate "forces in space and time in order to achieve a particular objective, in a situation of partial knowledge and where the enemy is also planning and reacting" (Lucek and Collander-Brown, 2017) and wrote a comprehensive reference paper detailing the use of AI in military wargames (Goodman, Sebastian and Lucas, 2020). The US Army has developed prototype wargaming software that uses AI to recommend Course of Actions improvements to commanders and staff (Schwartz et al, 2020), while US Navy researchers have conducted simulations to test various DRL algorithms' ability to train red team agents (Boron and Darken, 2020). In (Boron and Darken, 2020)'s experiment, a small group of mobile red team agents was allowed to learn attacking behavior using DRL against stationary blue team agents.

C. ALTERNATIVE APPROACHES

However, DRL is not the only option for developing intelligent wargaming agents. Evolutionary algorithms (EA), for example, begin with a (potentially random) selection of possible solutions and mutate the best solutions from that set in some fashion to form a new set of solutions. This process is repeated until a good solution, even if sub-optimal, is obtained. EAs are capable of moving in large random steps through the set of potential solutions and so are less prone to getting trapped in local optima than DNNs, which rely on derivatives and back-propagation. Genetic algorithms, simulated annealing, and CMA-ES (Hansen and Ostermeier, 2001) are examples of evolutionary algorithms.

As the number of possible actions in an environment grow larger, evaluating every possible solution quickly becomes infeasible. Statistical Forward Planning (SFP) (Perez, Samothrakis, Lucas and Rohlfshagen, 2013) is group of stochastic AI algorithms that do not require training. Instead, they use "Forward Models" to simulate possible outcomes from the current state and assign a value to the proposed action. While they do not require training data in advance, the computational burden of simulating possible outcomes every time a decision is required is expensive. Examples of SFPs are Monte Carlo Tree Search (MCTS) and Rolling Horizon Evolutionary Algorithms (RHEA). MCTS creates a tree of possible actions and calculates expected rewards at each action node in the current level, and then further expands the nodes with the highest expected reward to include subsequent action nodes. This process is continued until a solution is found. RHEA, on the other hand, uses EAs to generate a series of actions at each step of the simulation, and then takes the first action of the best series found as the next game step (Perez-Liebana et al, 2019).

Hybrid models also exist. Expert Iteration, for example, uses SFP to generate training data which is in turn used by supervised RL models to learn optimal policies (Anthony, Tian, and Barber, 2017). (Stanescu, Barriga, Hess and Buro, 2016) used Convolutional Neural Networks to evaluate sequences of actions during MCTS. (Khadkha and Tumer, 2018) proposed Evolutionary Reinforcement Learning as a hybrid algorithm that uses the population from EA to train a RL agent and then periodically reinserts the RL agent back into the EA population as a means of introducing gradient information back to the EA.

While DRL algorithms have achieved super-human performance in many games, they frequently exhibit non-human behavior. That is, taking actions that are counter-intuitive or actions that no rational human would choose in a given scenario. However, some algorithms have been developed in an attempt to inject a human-like quality into their behavior. Examples of such algorithms are training racing games to imitate human driving style (Muñoz, Gutierrez, and Sanchis, 2012) and Inverse Reinforcement Learning (Ng and Russell, 2000). Algorithms mimicking human behavior generally train a policy using supervised learning based on actual human gameplay. The goal of the algorithms is to minimize the error between a model's predicted actions and the actions actually taken by humans. Inverse Reinforcement Learning also uses human-generated data but rather than attempting to learn a policy directly from the data, it attempts to learn the reward function for the task that generated the data. Once the reward function has been learned, standard RL methods can then be used to learn a suitable policy for the agent to use.

D. SCALING ISSUES

Scalability refers to extending the use of an algorithm to an increasingly large environment with a larger number of agents, and potentially to a larger variety of agent types and possible actions from which agents can select. This is not a trivial extension as there is a combinatorial explosion of possible agent/action combinations in the larger environment. Assume that we have four homogeneous agents who can select from four possible actions. There are 256 possible agent/action combinations. If we extend the example to ten agents with ten possible actions, then there are 10 billion possible agent/action combinations to manage.

QT-Opt (Kalashnikov et al, 2018) uses a derivate-free optimization algorithm called the cross-entropy method to identify the action that maximizes a DNN's output. The algorithm maintains a buffer of labelled agent experiences from which in-training agents can sample. Thus, a large number of agents can be trained asynchronously by sampling from, and contributing to, this experience buffer.

(Salimans, Ho, Chen, Sidor and Sutskever, 2017) explored the use of natural evolution strategies (NES) as an alternative to RL-based techniques. NES works by using a statistical distribution to select possible solutions, which are in turn evaluated by a fitness function. NES then takes a step, based on the natural gradient, in the direction with the highest expected fitness value. The process repeats until some stopping criteria is met. While requiring more data than RL-based methods, NES was observed to require significantly less computing power since they lack value functions and do not perform back-propagation. Further, multiple agents can be deployed in parallel with little dependence, making the process extremely scalable.

By adding heuristic/expert knowledge to the AlphaZero RL framework, (Moy and Shekh, 2019) was able to both train a DRL model that could outperform the heuristics used to train it and realized significant computational savings over using MCTS alone.

Another potential avenue for increased scalability is hierarchical models. (Wang et al, 2020) proposed a hierarchical RL algorithm. The higher-level network selects actions (move, fire, etc.) while the lower-level network chooses the direction. In the authors' determination, the lower-lever policies for agents are relatively independent and cooperation is more a function of the higher-level network decisions. Therefore, once a lower-level network solution is found for one agent, it can be freely copied to other agents. Alternatively, the lower-level network can simply be replaced with heuristic methods or models learned offline.

(Pentreath, 2020) presented four opportunities to improve scalability: developing innovate computing architecture for low-resource environments, model compression techniques such as quantization where model parameters, etc. are stored in a lower bit state, making models smaller by removing weights that have little impact on prediction, and model distillation in which smaller neural networks are taught by a larger neural network to behave like the larger neural network.

Open source distributed computing libraries, such as Uber's Fiber (Zhi, Wang, Clune and Stanley, 2020) and DeepMind's Acme (Hoffman et al, 2020), have been developed to facilitate distributed reinforcement learning. Solutions such as these allow developers to dynamically scale up algorithms to meeting existing available resources with limited coding overhead.

II. OBJECTIVES

This research intends to evaluate the viability of using reinforcement learning to train competent adversarial forces. We will seek to answer the following questions:

- Is there a significant, quantifiable difference in the various reinforcement learning algorithms' abilities to train adversaries capable of defeating friendly forces?

- Is there a quantifiable tradeoff between a reinforcement learning algorithm's ability to train adversaries capable of defeating friendly forces, and the time/resources required to train the adversary?

- Given an adversary that has been reinforcement learning-trained to defeat friendly forces, how sensitive are the adversary's capabilities to subsequent changes in friendly force strategy and/or simulation environment?

- How do the answers to the above questions affect the ability to scale the simulations?

III. METHODOLOGY

A. EXPERIMENT DESIGN

"Good wargames are small and have an aggressive and dynamic red team."

(Pournelle, 2017, p.52)

The goal of this experiment is to identify the effect the changes in simulation environment and other training-specific variables have on the time required to train a red team to defeat a blue team and to test the resiliency of the trained red team by subsequently modifying wargame conditions. To accomplish this, the experiment was conducted in three stages.

1. Stage 1

A mobile blue team is trained to defeat a stationary red team, similar to the experiment conducted by (Boron, 2020). This learned blue strategy serves as a proxy for a real-world combat strategy for which we want to develop a red team capable of defeating.

2. Stage 2

The wargame is reinitialized with the blue team's strategy fixed at that learned in Stage 1 and the red team is trained to defeat the blue team in the current stage. The red team is now capable of defeating blue under its current strategy. Information on the time required to train the red team to defeat a blue team using a fixed strategy is collected here.

3. Stage 3

Having trained a competent red team in Stage 2, the resiliency of the trained red team is tested in the current stage by injecting random movement into the blue team's previous strategy. 100 simulated battles were conducted between the trained red team and the blue team with a modified strategy, and information on the proportion of wargames in which the red team was able to defeat the blue team is collected here.

In each stage, the blue and red team began on opposing ends of a hexagonal map, as illustrated in Figure 1.

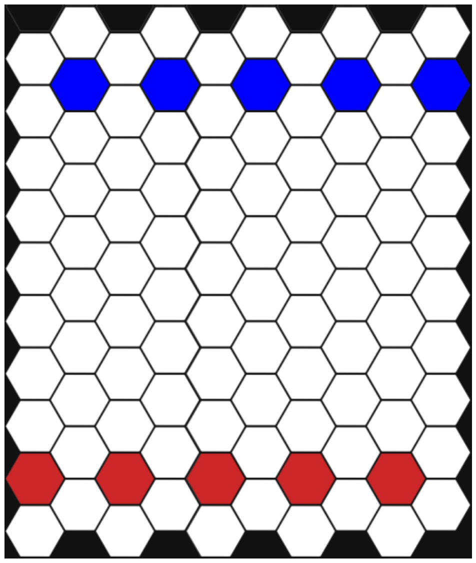

Figure 1. Example environment with opposing red and blue agents

B. EXPERIMENT VARIABLES

The following variables are set at the start of Stage 1 and remain constant through the subsequent stages. This experiment was developed as a screening design using (JMP, n.d.), designed to test the effects of up to two-way interactions. Further, the experiment was repeated for each of three RL algorithms described below, and two repetitions were taken at each combination of variable settings. When possible, variable names and settings were selected to remain consistent with previous research (Boron, 2020) and such that the experiment could be completed within the time allotted for this research. Evaluated settings are shown in brackets.

1. Map Width: {10, 15}

Each simulation was completed on a map of *n* x *n* hexagons. Red and blue teams were placed in a row on opposite ends of the map at the start of each simulation, with each individual agent of the team occupying its own hexagon. Agents can move in any of six directions from a given hexagon but are not allowed to move outside a map's boundaries.

2. Red Team Size: {2, 5}

The number of red team agents at the start of the simulation. Each individual agent was identical in capabilities, namely, to move from hexagon to hexagon in permitted directions, to opt to remain in place, and to attack enemy agents when in range.

3. Blue Team Size: {2, 5}

The number of blue team agents at the start of the simulation. Each individual agent was identical in capabilities, namely, to move from hexagon to hexagon in permitted directions, to opt to remain in place, and to attack enemy agents when in range.

4. Training Duration: {50, 250}

The number of epochs. Equivalent to the number of policy updates.

5. Learning Factor: {0.965, 0.995}

The discount factor. Determines the timeframe over which agents place most value when evaluating potential moves. Larger learning factors place a greater value on long-term gains while smaller learning factors place a greater value on short-term gains.

6. Learning Rate: {0.0001, 0.01}

The learning rate for the optimization algorithm. Larger learning rates allow for larger changes in the DRL network's parameters at each update and thereby more opportunity to explore vastly different solutions. Smaller learning rates restrict the amount by which the parameters can be changed at each update, allowing the algorithm more time to linger in and exploit areas in which potentially superior solutions may exist.

7. Blue Alpha: {0.25, 0.75}

The probability of a blue team agent making a uniformly random-selected move in Stage 3. Conversely, the blue team agents select moves according to the strategy learned in Stage 1 with probability (1 – Blue Alpha).

8. NN Structure: {(32,), (96,96)}

The structure of the neural network to be optimized through simulation. Networks consisted of either a single 32-node hidden layer or two 96-node hidden layers. All networks were fully connected and used Rectified Linear Units (ReLu) activation functions.

9. Combat Model: {Deterministic, Stochastic}

The method of adjudicating damage from combat between two opposing agents. The amount of damage an attacking force could deliver was calculated as:

Deterministic Lanchester:	combat efficiency * attacking force size
Stochastic Lanchester:	combat efficiency * attacking force size * RAND,

where RAND is a random float selected uniformly from the (0, 1) interval.

Both sides started with a force size of 150, intended to simulate a company-sized unit. Combat efficiency was set at 0.1. See (Boron, 2020, pp.26-28) for details on how combat is adjudicated using these equations.

10. Delta: {0.001, 0.05}

This variable was only applicable to TRPO and limits the difference between the current policy and a new candidate policy selected by the optimizer. Smaller values constrain the algorithm to focus on candidate policies that are similar to the current solution policy, while larger values allow the algorithm to test candidate policies that are more dissimilar to the current solution.

Any variables not included here remained at their default settings available at (Spinning Up, n.d.).

C. CODE AND ALGORITHMS

The experiment was created using Python3 in (Anaconda, n.d.), with particular use of the (Gym, n.d.) and (Spinning Up, n.d.) packages. The code was based on that developed for, and generously shared by, (Boron, 2020) and (Boron and Darken, 2020). Gym is a toolkit for developing and testing reinforcement learning algorithms. SpinningUp provides a framework for developing custom reinforcement learning environments and experiments. SpinningUp also provides several off-the-shelf DRL algorithms for use in experimentation, as well as tools for recording key training data and outputting visual results. The algorithms tested in this experiment, described below, were selected because they were readily available in SpinningUp, with minimum modification required, and represent an evolution in the development of RL algorithms (SpinningUp, *Why These Algorithms?* n.d.). Further, these algorithms were used in previous research in a similar application (Boron, 2020). Descriptions and quoted text are taken from (Spinning Up, n.d.). and detailed information can be found there:

1. Vanilla Policy Gradient (VPG)

An "on-policy algorithm", VPG updates policy parameters via stochastic gradient ascent in an attempt to maximize the finite-horizon return of the policy. "It explores by sampling actions according to the latest version of its stochastic policy…Over the course of training, the policy typically becomes progressively less random, as the update rule encourages it to exploit rewards that it has already found" (Spinning Up, n.d.). Probabilities of actions that lead to higher returns are pushed up and probabilities of actions that lead to lower return are pushed down until you arrive at the optimal policy.

2. Trust Region Policy Optimization (TRPO)

In normal policy gradient algorithms, such as VPG, changes are made to the policy parameters such that old and new policies do not diverge greatly over a single update. TRPO, however, "updates policies by taking the largest step possible to improve performance, while satisfying a special constraint on how close the new and old policies are allowed to be" (Spinning Up, n.d.).

3. Proximal Policy Optimization (PPO)

Like TRPO, PPO seeks to take the biggest possible step to improve policy, "without stepping so far that we accidentally cause performance collapse" (Spinning Up, n.d.). PPO employs first-order methods, while TRPO's methods are second-order, making PPO simpler to implement.

D. ANALYSIS

Once each simulation was completed, the total time, measured in seconds, required to train the red team in Stage 2 (“Training Time”) and the proportion of simulations in which the red team was able to completely eliminate the blue team in Stage 3 (“Win Rate”) were calculated. First the data for Training Time and Win Rate was analyzed to get a sense of the distribution and any possible relationships of note. Then, linear models were built to identify any variables and their interactions that may have a significant effect on Training Time and/or Win Rate. Models were built and analyzed from both with and without respect to individual DRL algorithm.

1. Descriptive Statistics

TRPO required the most time to train the red team, on average, of any algorithm. TRPO also had the smallest standard error due to the presence of the additional Delta variable requiring more simulations in the screening design. VPG, on the other hand required the least amount of time and had the smallest standard error.

Table 1. Training Time (seconds)

	ALL	PPO	TRPO	VPG
Mean	577.1841	567.6558	603.0471	552.9781
Std Dev	393.2262	383.2536	412.856	378.8281
Std Err Mean	22.55307	39.95695	37.68843	39.49556
Upper 95% Mean	621.5646	647.0254	677.6739	631.4312
Lower 95% Mean	532.8036	488.2862	528.4202	474.5251
N	304	92	120	92

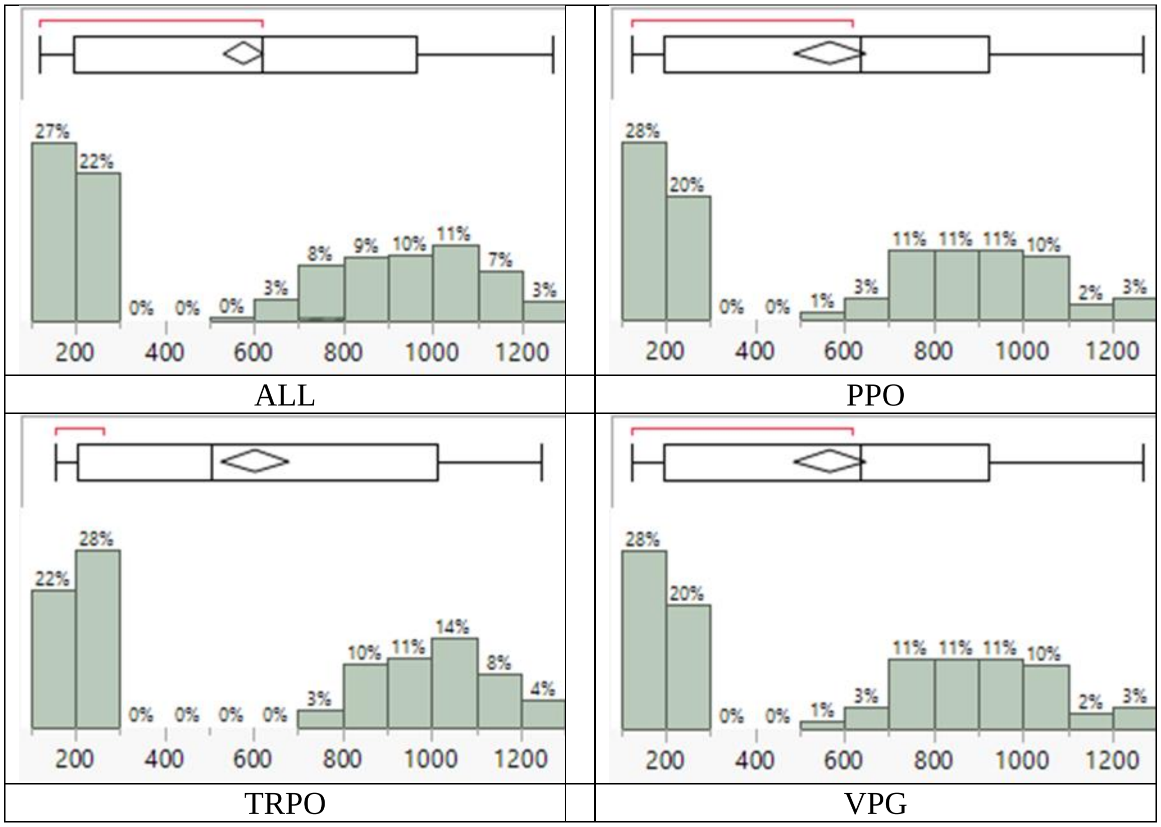

Figure 2. Histograms of training time (in seconds)

As can be seen from Figure 2, Training Times were bimodal whether considered together or on a by-algorithm basis. In either case, nearly 50% of simulations were concluded within 300 seconds. With the exception of VPG, the remaining simulation Training Times were distributed approximately symmetrically in a bell-shape around a mean ranging from 900 to 1000 seconds.

Table 2. Win Rate

	All	PPO	TRPO	VPG
Mean	0.321149	0.300066	0.412328	0.223302
Std Dev	0.431033	0.433981	0.452628	0.375869
Std Err Mean	0.024721	0.045246	0.041319	0.039187
Upper 95% Mean	0.369796	0.389941	0.494144	0.301142
Lower 95% Mean	0.272501	0.210191	0.330512	0.145462

N	304	92	120	92
N Zero	180	57	59	64
Proportion of simulations which neither side won	0.592105	0.619565	0.491667	0.695652

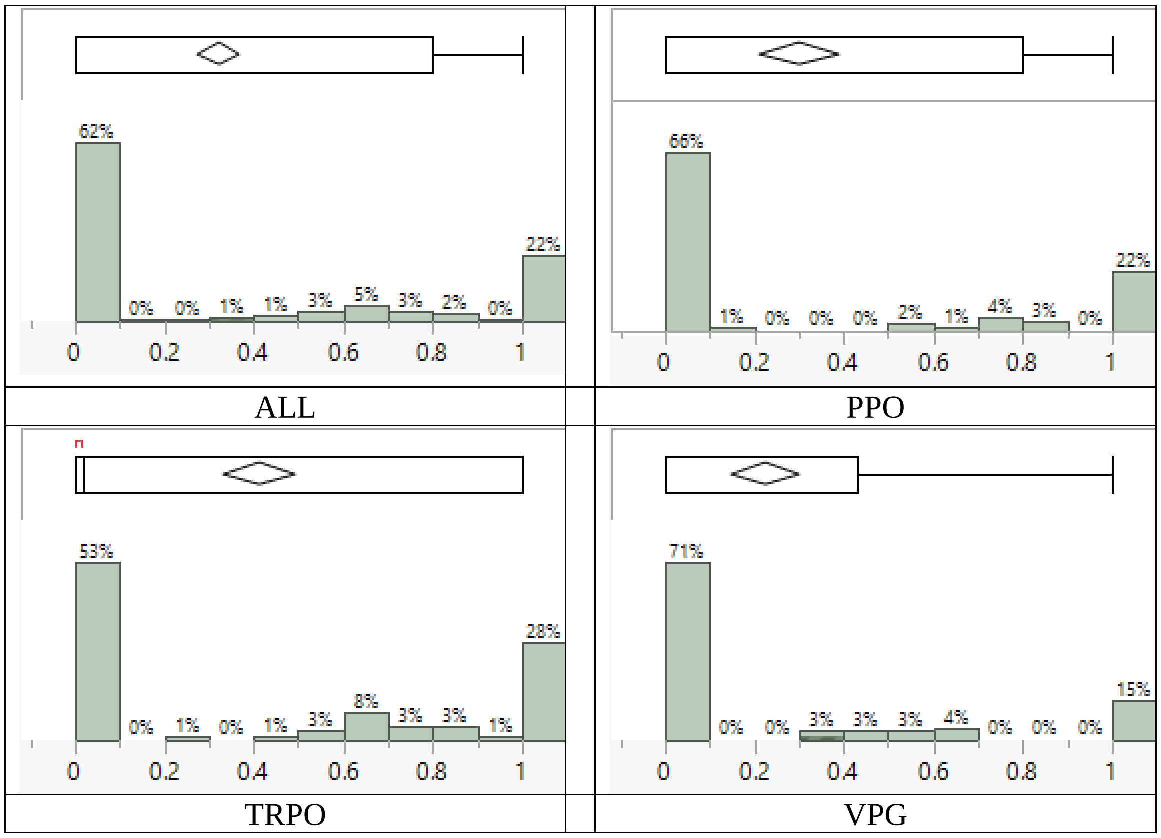

Figure 3. Histograms of win rate

Figure 3 indicates that Win Rate is also bimodal, with a minority of outcomes disperses between the two extremes. In every case, the majority of simulations ended with no wins, meaning that neither side was completely able to eliminate the other. The specific rate of no wins ranged from 59% to 70% as show in Table 2. The percentage of simulations ending in wins ranged from 15% to 28%, with the remaining outcomes distributed in an approximate bell-shape between the two extremes.

Full Correlation Matrices are Available in Table 12 of the Appendix. Training Time was strongly positively correlated with Training Duration when evaluating all algorithms together or separately. The correlation coefficient ranged from 0.94 to 0.97, each with an associated p-value of <0.0001 [Table 13 of the Appendix]. Training Time in PPO and VPG had smaller, but statistically significant, correlations with Blue Team Size of 0.23 and 0.25, and p-values of 0.03 and 0.02, respectively.

Win Rate was strongly negatively correlated with Blue Team Size when evaluating all algorithms together or separately. The correlation coefficient ranged from -0.56 to -0.815, each with an associated p-value of <0.0001. Win Rate in PPO and VPG had smaller, but statistically significant, correlations with Map Width of -0.26 and -0.25, and p-values of 0.01 and 0.02, respectively. TRPO's Win Rate had a moderate positive correlation of 0.27 with Red Team Size and p-value of 0.003.

Linear models were built to evaluate the effects of the previously enumerated experiment variables on Training Time and Win Rate. The models evaluated took the form:

$$Y_i = \beta_0 + \sum_{j=1}^{m} \beta_j X_{ij} + \sum_{j=1}^{m} \sum_{k \neq j}^{m} \beta_{jk} X_{ij} X_{ik} + \varepsilon_i, \quad i = 1, \dots, n$$

where Y_i is the predicted value at the ith observation of the m variables

X_{ij} is ith observation of the jth variable

β_0 is the intercept term

β_j is the coefficient for the jth variable

β_{jk} is the coefficient for the interaction between the jth and kth variables

ε_i is the random error term, assumed to be Normally distributed with a mean of zero and constant variance

n is the total number of observations

2. Training Time

First a linear model that did not consider a difference in algorithm was evaluated. That is, the data from all three algorithms were treated as if they had been generated from a single source. However, since the Delta variable is only used in the TRPO algorithm, this model was evaluated using all variables except Delta. The variable Blue Alpha was also not considered in any Training Time model since Training Time is calculated in Stage 2 and Blue Alpha is only relevant to Stage 3.

The model was significant (p=0.0001) and is included in Table 3. All individual variables had a significant effect on Training Time, with the exceptions of Learning Factor and Combat Model. The effects of individual variables on Training Time were all positive, except for Learning Rate and NN Structure. Training Duration had the largest positive effect on Training Time, far eclipsing the effect of other individual variables. This is expected as it is reasonable for Training Time to increase along with the number of simulations. The next largest positive significant effect came from Blue Team Size, much larger than the remaining individual variables. This is also reasonable as it will take the red team longer to learn how to defeat an increasingly larger blue team.

NN Structure and Learning Rate were the only variables with significant negative effect on Training Time. Learning Rate's negative effect is not surprising as a larger Learning Rate allows the optimizer to search for a broader range of solutions. Not all of these solutions will be good and so it can take longer for the optimizer to home in on a promising set of solutions. The negative effect of NN Structure, however, is surprising. The experiment design does not allow differentiation between the effects of increased number of nodes and increased number of layers. However, as both of these numbers increase, the overall Training Time decreased significantly. Perhaps there is an efficiency gained by using a larger network structure, in terms of the time required to reach a solution, that outpaces the increased time required for the optimizer to train the increased number of parameters in a larger network. While more research would be needed to reach a definitive conclusion, it is not reasonable to expect increasingly larger networks to have a continuing negative effect on Training Time. The increased number of parameters requiring training would likely cause NN Structure to have a positive effective on Training Time once the network has reached a critical size, in terms of the number of hidden layers and/or the number of nodes.

Several two-way interactions were also significant, but most of these were in line with expectations given the effects of individual variables. One notable exception was the interaction between Training Duration and NN Structure, which was negative. Individually, Training Duration had a very large positive effect on Training Time while NN Structure had a smaller negative effect. A simultaneous increase in both of these variables however has an overall negative effect, lending credence to the previous suggestion of efficiency gains from a larger NN Structure. However, the same caveat from earlier applies. It is unlikely that indefinite increases in NN Structure, paired with increasing long Training Durations, will continues to have a negative effect on Training Time.

The interaction between Red and Blue Team Sizes was also negative. While these might seem counter-intuitive at first glance given that both of the individual variables had positive effects, we hypothesize that simultaneously increasing both variables also makes it easier for opposing agents to find each other on the simulated battlefield and engage in combat, thus reducing the overall Training Time.

Table 3. Training Time ALL Model

Term	Estimate	Std Error	Prob>|t|	Significance
Intercept	566.7882	3.549825	<.0001	***
Map Width	19.30811	3.64956	<.0001	***
Red Team Size	18.79764	3.735498	<.0001	***
Blue Team Size	73.10628	3.67727	<.0001	***
Training Duration	375.7265	3.804612	<.0001	***
Learning Factor	3.079145	3.725537	0.4093	
Learning Rate	-6.64219	3.745186	0.0773	*
NN Structure[(32,)]	-21.8369	3.687565	<.0001	***
Combat Model[Deterministic]	1.928856	3.778603	0.6101	
Map Width*Red Team Size	-3.58364	3.689783	0.3323	
Map Width*Blue Team Size	-5.32753	3.623559	0.1427	
Map Width*Training Duration	13.12429	3.681089	0.0004	***
Map Width*Learning Factor	2.092978	3.795309	0.5818	
Map Width*Learning Rate	-4.33738	3.826685	0.258	
Map Width*NN Structure[(32,)]	-2.70593	3.723258	0.468	
Map Width*Combat Model[Deterministic]	0.981867	3.880303	0.8004	
Red Team Size*Blue Team Size	-11.968	3.695362	0.0014	***
Red Team Size*Training Duration	10.7236	3.724014	0.0043	***
Red Team Size*Learning Factor	-4.39013	3.719705	0.239	

Red Team Size*Learning Rate	-5.50414	3.627005	0.1303	
Red Team Size*NN Structure[(32,)]	3.495108	3.755447	0.3529	
Red Team Size*Combat Model[Deterministic]	-1.08664	3.78111	0.774	
Blue Team Size*Training Duration	48.45899	3.711667	<.0001	***
Blue Team Size*Learning Factor	4.646716	3.690479	0.2091	
Blue Team Size*Learning Rate	-2.24097	3.657138	0.5406	
Blue Team Size*NN Structure[(32,)]	-6.03209	3.762882	0.1101	
Blue Team Size*Combat Model[Deterministic]	2.160163	3.695079	0.5593	
Training Duration*Learning Factor	5.959679	3.750948	0.1133	
Training Duration*Learning Rate	-5.3535	3.798781	0.1599	
Training Duration*NN Structure[(32,)]	-19.3901	3.763516	<.0001	***
Training Duration*Combat Model[Deterministic]	1.383407	3.794258	0.7157	
Learning Factor*Learning Rate	0.155986	3.947207	0.9685	
Learning Factor*NN Structure[(32,)]	0.789762	3.663633	0.8295	
Learning Factor*Combat Model[Deterministic]	2.497323	3.905414	0.5231	
Learning Rate*NN Structure[(32,)]	-2.64627	3.732624	0.479	
Learning Rate*Combat Model[Deterministic]	-4.13691	3.931581	0.2936	
NN Structure[(32,)]*Combat Model[Deterministic]	-1.8863	3.785331	0.6187	

(Significance: *** < 0.01, **<.05, *<.10)

Next, Linear models were built for each of the three algorithms evaluated. It is instructive to observe which variables, if any, have significant effects across all three algorithms and which differ depending on the algorithm selected. The full results are too large to include here but are available in the Appendix. Table 4 summarizes the single variables and two-way interactions that had significant effects on Training Time.

Some variables consistently affected Training Time regardless of algorithm such as Map Width, Blue Team Size and Training Duration, while others were algorithm specific. NN Structure's effect on Training Time, for example, was significant but only when using PPO and VPG, not for TRPO. An increase in Red Team only increased Training Time in the TRPO algorithm and had little effect on the other algorithms. Similar to the model where all algorithms are considered together, increased Learning Rate had a negative effect on Training Time but only for the PPO algorithm. And Combat Model, which was not significant when all algorithms were considered together, had a significant effect on the Training Time required for the PPO algorithm but not for TRPO and VPG.

The interactions of Map Width and Training Duration, and Blue Team Size and Training Duration, were significantly positive across all three algorithms. Several interactions were significant across only two algorithms. Training Duration and NN Structure had a significantly negative effect on Training Time in PPO and VPG but did not have a significant effect under TRPO. This is consistent with the significances observed in these two individual variables among the three algorithms. The interaction of Red and Blue Team sizes was significantly negative for PPO and VPG, much like the combined model, but not significant under TRPO. While there were interactions that were significant for both TRPO and VPG, such as Red Team Size and Training Duration, there were no two-way interactions that were significant for both PPO and TRPO but not VPG.

Table 4. Training Time Models By-Algorithm

	PPO		VPG		TRPO	
Term	Est.	Sig.	Est.	Sig.	Est.	Sig.
Intercept	548.63	***	535.88	***	604.27	***
Map Width	27.35	***	17.13	***	16.24	***
Red Team Size	4.90		2.50		36.19	***
Blue Team Size	75.93	***	88.67	***	62.77	***
Training Duration	358.36	***	352.08	***	398.52	***
Learning Rate	-15.09	**	-0.66		-1.26	
NN Structure[(32,)]	-26.78	***	-35.61	***	-3.74	
Combat Model[Deterministic]	12.00	*	-0.87		2.66	
Map Width*Training Duration	20.68	***	8.59	***	13.82	***
Map Width*Learning Rate	-9.93		-10.80	***	3.73	
Map Width*NN Structure[(32,)]	7.50		4.60	**	-2.09	
Map Width*Combat Model[Deterministic]	-2.20		-7.36	***	-4.93	*
Red Team Size*Blue Team Size	-12.10	*	-20.00	***	-1.71	
Red Team Size*Training Duration	3.31		5.18	**	20.81	***
Red Team Size*NN Structure[(32,)]	7.28		5.59	***	-3.13	
Red Team Size*Combat Model[Deterministic]	-1.09		4.64	**	1.74	
Blue Team Size*Training Duration	52.27	***	60.64	***	42.17	***
Blue Team Size*Learning Rate	-6.40		3.95	**	-0.88	
Blue Team Size*NN Structure[(32,)]	-6.05		-9.25	***	0.05	
Blue Team Size*Combat Model[Deterministic]	-1.58		-1.71		5.13	*
Training Duration*Learning Factor	15.04	**	1.94		1.85	
Training Duration*NN Structure[(32,)]	-18.76	***	-22.82	***	1.57	
Training Duration*Combat Model[Deterministic]	13.87	**	3.46	*	1.80	
Learning Factor*Learning Rate	-2.32		-5.34	**	-7.66	**
Learning Factor*Combat Model[Deterministic]	1.36		-5.22	**	4.05	
Learning Rate*NN Structure[(32,)]	2.26		-9.18	***	-1.65	
Learning Rate*Combat Model[Deterministic]	6.04		6.09	***	-4.75	*
NN Structure[(32,)]*Combat Model[Deterministic]	-6.14		-1.43		0.54	
Red Team Size*Delta					-5.31	**
Delta*NN Structure[(32,)]					-4.86	*

(Significance: *** < 0.01, **<.05, *<.10)

3. Win Rate

As with Training Time, a model that did not consider a difference in algorithm was first evaluated on Win Rate, excluding the variable Delta. Unlike the Training Time models however, the variable Blue Alpha was considered in all Win Rate models since it was a factor in Stage 3 where Win Rate was calculated.

The constructed model was significant (p=0.0001) and is included in Table 5. Map Width and Combat Model had a significant effect on Win Rate, while Training Duration did not. Both Map Width and Combat Model both have a negative effect on Win Rate, albeit minor. As before, Red and Blue Team Sizes were both significant, with larger Red Team Size contributing to a higher Win Rate and larger Blue Team Size contributing to a lower Win Rate.

There were several significant interactions but most of these did not have large effects. Of note, there are several significant interactions including Learning Factor that had a significant effect on Win Rate, whereas none had an effect on Training Time. Two of these newly significant interactions were between Learning Factor and Blue Alpha, and Learning Factor and Learning Rate. While none of the factors in these combinations were significant on their own, combined they had a significant effect on Win Rate.

Table 5. Win Rate ALL Model

Term	Estimate	Std Error	Prob>|t|	Significance
Intercept	0.312439	0.016375	<.0001	***
Map Width	-0.06522	0.016916	0.0001	***
Red Team Size	0.054481	0.017522	0.0021	***
Blue Team Size	-0.32224	0.017574	<.0001	***
Training Duration	0.008892	0.01831	0.6276	
Learning Factor	-0.00781	0.017368	0.6534	
Learning Rate	0.001881	0.017385	0.9139	
Blue Alpha	-0.00826	0.017463	0.6368	
NN Structure[(32,)]	0.019773	0.01772	0.2655	
Combat Model[Deterministic]	-0.03641	0.017847	0.0424	**
Map Width*Red Team Size	0.015809	0.017673	0.3719	
Map Width*Blue Team Size	0.0603	0.017239	0.0006	***
Map Width*Training Duration	0.062047	0.01767	0.0005	***
Map Width*Learning Factor	-0.01491	0.018047	0.4096	
Map Width*Learning Rate	0.040192	0.017989	0.0263	**
Map Width*Blue Alpha	0.00621	0.017712	0.7262	
Map Width*NN Structure[(32,)]	-0.00441	0.017355	0.7994	
Map Width*Combat Model[Deterministic]	-0.03803	0.01822	0.0379	**
Red Team Size*Blue Team Size	-0.06306	0.017466	0.0004	***
Red Team Size*Training Duration	0.010771	0.017885	0.5476	
Red Team Size*Learning Factor	-0.03011	0.018059	0.0966	*
Red Team Size*Learning Rate	-0.02336	0.01703	0.1713	
Red Team Size*Blue Alpha	-0.02627	0.017627	0.1373	
Red Team Size*NN Structure[(32,)]	0.057643	0.017418	0.0011	***
Red Team Size*Combat Model[Deterministic]	-0.00242	0.017789	0.8917	
Blue Team Size*Training Duration	-0.00653	0.017359	0.7073	
Blue Team Size*Learning Factor	0.014409	0.017476	0.4104	

Blue Team Size*Learning Rate	0.020102	0.017189	0.2433	
Blue Team Size*Blue Alpha	-0.01322	0.017794	0.4583	
Blue Team Size*NN Structure[(32,)]	-0.01682	0.017909	0.3485	
Blue Team Size*Combat Model[Deterministic]	0.010371	0.017095	0.5446	
Training Duration*Learning Factor	-0.02321	0.017799	0.1935	
Training Duration*Learning Rate	-0.01467	0.017791	0.4104	
Training Duration*Blue Alpha	0.004449	0.01824	0.8075	
Training Duration*NN Structure[(32,)]	0.024922	0.017495	0.1555	
Training Duration*Combat Model[Deterministic]	0.031311	0.018055	0.0841	*
Learning Factor*Learning Rate	-0.03466	0.018188	0.0578	*
Learning Factor*Blue Alpha	0.03702	0.017539	0.0358	**
Learning Factor*NN Structure	-0.00755	0.016976	0.6567	
Learning Factor*Combat Model[Deterministic]	0.033734	0.018199	0.0649	*
Learning Rate*Blue Alpha	0.001714	0.017195	0.9207	
Learning Rate*NN Structure[(32,)]	0.016066	0.017308	0.3542	
Learning Rate*Combat Model[Deterministic]	-0.01599	0.018593	0.3906	
Blue Alpha*NN Structure[(32,)]	0.00554	0.017149	0.7469	
Blue Alpha*Combat Model[Deterministic]	-0.01329	0.017121	0.4382	
NN Structure[(32,)]*Combat Model[Deterministic]	0.026434	0.017553	0.1333	

(Significance: *** < 0.01, **<.05, *<.10)

As with Training Time, we also built linear models to evaluate the effect of individual variables and their interactions on win rate by algorithm. Table 6 presents a summary of these results with the full output available in the Appendix. Surprisingly, Training Duration was the only single variable to be significant across all three algorithms, although the signs of the effects differed. Combat Model was significant with large positive effects under PPO and VPG but otherwise, none of the single variables were significant in more than one algorithm. Red Team Size and Blue Team Size only had a significant effect on the Win Rate when using the TRPO algorithm, while Map Width, Blue Alpha and NN Structure only affected Win Rate under PPO. However, the effect size of these variables under PPO was relatively large.

Of the two-way interactions, only Map Width and Blue Team Size, and Training Duration and Combat Model, had a significant effect on Win Rate over all three algorithms. While Learning Rate was only significant in VPG, Learning Rate had significant interactions with several variables common to VPG and TRPO. Namely, Map Width, Red Team Size and Learning Factor. Combat Model, on the other hand, had two significant, large interaction effects with two variables common to PPO and VPG: Learning Factor and Learning Rate.

Table 6. Win Rate Models By-Algorithm

	PPO		VPG		TRPO	
Term	Est.	Sig.	Est.	Sig.	Est.	Sig.
Intercept	-0.18		-0.17		0.38	***
Map Width	0.44	**	0.10		-0.03	
Red Team Size	0.14		0.19		0.11	***
Blue Team Size	0.02		-0.19		-0.39	***
Training Duration	-0.20	**	-0.17	***	0.08	***
Learning Factor	-0.27		0.06		0.01	
Learning Rate	0.08		-0.24	*	0.02	
Blue Alpha	-0.73	***	0.05		-0.02	
NN Structure[(32,)]	0.58	***	0.08		-0.03	
Combat Model[Deterministic]	0.48	**	0.31	**	-0.03	
Map Width*Red Team Size	0.02		-0.03		0.00	
Map Width*Blue Team Size	0.10	***	0.13	***	0.04	*
Map Width*Training Duration	0.22	**	0.07		0.09	***
Map Width*Learning Factor	0.04		-0.02		-0.02	
Map Width*Learning Rate	-0.01		0.16	***	0.06	**
Map Width*Blue Alpha	0.01		-0.04		0.01	
Map Width*NN Structure[(32,)]	0.10	***	-0.01		0.00	
Map Width*Combat Model[Deterministic]	-0.10	**	0.03		-0.05	**
Red Team Size*Blue Team Size	-0.08	**	-0.03		-0.06	**
Red Team Size*Training Duration	0.08		0.05		0.02	
Red Team Size*Learning Factor	-0.04		0.02		-0.03	
Red Team Size*Learning Rate	-0.03		-0.05	**	0.05	**
Red Team Size*Blue Alpha	-0.05		0.01		-0.04	
Red Team Size*NN Structure[(32,)]	0.02		0.13	***	0.02	
Red Team Size*Combat Model[Deterministic]	0.05		0.03		-0.01	
Blue Team Size*Training Duration	0.15	*	0.03		-0.05	**
Blue Team Size*Learning Factor	0.00		0.01		-0.02	
Blue Team Size*Learning Rate	0.04		-0.03		0.00	
Blue Team Size*Blue Alpha	-0.01		0.04		0.01	
Blue Team Size*NN Structure[(32,)]	0.01		-0.07	**	0.01	
Blue Team Size*Combat Model[Deterministic]	0.03		0.00		0.01	
Training Duration*Learning Factor	-0.12		0.05		-0.03	
Training Duration*Learning Rate	0.06		-0.11	*	-0.03	
Training Duration*Blue Alpha	-0.30	***	0.03		0.02	
Training Duration*NN Structure[(32,)]	0.23	***	0.00		0.05	**
Training Duration*Combat Model[Deterministic]	0.19	**	0.16	**	0.06	***
Learning Factor*Learning Rate	-0.02		-0.07	**	-0.06	***
Learning Factor*Blue Alpha	0.03		0.07	***	0.00	
Learning Factor*NN Structure[(32,)]	-0.03		0.03		0.01	
Learning Factor*Combat Model[Deterministic]	0.11	***	0.06	**	0.03	
Learning Rate*Blue Alpha	0.03		-0.04		0.02	
Learning Rate*NN Structure[(32,)]	-0.02		0.08	***	0.04	
Learning Rate*Combat Model[Deterministic]	0.12	***	-0.08	**	0.02	
Blue Alpha*NN Structure[(32,)]	0.03		-0.01		0.02	
Blue Alpha*Combat Model[Deterministic]	-0.06	*	0.00		0.00	
NN Structure[(32,)]*Combat Model[Deterministic]	0.08	**	0.02		-0.03	
Delta					0.02	
Map Width*Delta					0.02	
Red Team Size*Delta					0.03	
Blue Team Size*Delta					-0.04	*
Training Duration*Delta					0.00	
Learning Factor*Delta					-0.03	
Learning Rate*Delta					0.01	
Blue Alpha*Delta					-0.01	
Delta*NN Structure[(32,)]					-0.02	
Delta*Combat Model[Deterministic]					0.02	

(Significance: *** < 0.01, **<.05, *<.10)

4. Difference in Means

Next, we examine whether the algorithm used had a significant effect on the mean Training Time and/or Win Rate and if so, were there significant differences in the performance of the individual algorithms. To do this, we performed an ANOVA test on the algorithms against Training Time and Win Rate. If the ANOVA results were found to be significant, we followed up by simultaneously testing pairs of algorithms to determine if the mean performance of one was better than others. A straightforward t-test of significance between each pair of algorithms would inflate the risk of a type 1 error, or mistakenly concluding a significant difference exists when there is none. To counter this, we employ Tukey's Honest Significant Difference Test to test for significant differences in performance between pairs of algorithms. We also use a Tukey-Kramer adjustment to account for the different sample sizes between algorithms.

Table 7. ANOVA for Training Time Model

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	2	142525	71263	0.4592
Error	301	46709413	155181	Prob > F
C. Total	303	46851938		0.6322

The p-value of 0.6322 indicates that there is no significant difference in the mean Training Time between the algorithms examined here. Therefore, we do not conduct any follow-up tests.

Table 8. ANOVA for Win Rate Model

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	2	1.919339	0.959669	5.3124
Error	301	54.374916	0.180648	Prob > F
C. Total	303	56.294255		0.0054

In contrast to Training Time, the p-value of 0.0054 indicates a strong likelihood that their does exist a difference in mean Win Rate between the algorithms examined here. Next, we use Tukey's Test to examine differences in the algorithms mean effect on Win Rate on a pair-wise basis.

Table 9. Tukey's Test for Win Rate Model

Algorithm 1	Algorithm 2	Difference	Std Error	t Ratio	Prob>|t|	Lower 95%	Upper 95%
PPO	TRPO	-0.112262	0.058898	-1.91	0.1387	-0.250992	0.0264684
PPO	VPG	0.076764	0.062667	1.22	0.4394	-0.070844	0.2243719
TRPO	VPG	0.189026	0.058898	3.21	0.0042	0.050296	0.3277564
Quantile = 1.96788		DF = 301.0					

The test indicates that there is no significant difference in the mean Win Rate between PPO and TRPO, nor between PPO and VPG. However, the p-value of 0.0042 indicates that there is a statistically significant difference in the mean Win Rates of TRPO and VPG. It can be observed from Table 2 that the mean Win Rate of TRPO is 0.4123. Compared to the mean Win Rates of PPO and VPG, 0.3000 and 0.2233 respectively, we can conclude that TRPO produces, on average, higher win rates than PPO or VPG.

5. Relationship Between Training Time and Win Rate

We have thus far examined the relationship between several independent variables and the dependent variables Training Time and Win Rate. Finally, we examine the relationship between the two dependent variables. The following correlations are extracted from Table 12 and Table 13:

Table 10. Pearson Correlation Between Training Time and Win Rate

	Correlation	p-value
Combined	-0.080	0.1650
PPO	-0.209	0.0457

TRPO	0.025	0.7850
VPG	-0.156	0.1383

The correlation coefficients generally indicate a negative relationship between Training Time and Win Rate. While this is also supported by the previous general observations made under the full and by-algorithm linear models that Win Rate tends to decrease as Training Duration increases, with the exception of TRPO, the significances of the correlations are weak. The correlation p-values in Table 10 are also the significance values for simple linear regression models relating Training Time to Win Rate. The regression output for PPO, the only significant algorithm, is presented in Table 11.

Table 11. Simple Linear Regression Relating Training Time and Win Rate under PPO

Term	Estimate	Std Error	Prob>|t|
Intercept	0.4343085	0.079814	<.0001
Training Time	-0.000236	0.000117	0.0457

Although Training Time does have a statistically significant effect on Win Rate when training under the PPO algorithm, the actual effect -0.000236 is very small.

IV. DISCUSSION

It would be ideal to identify variables that simultaneously reduce training time and increase win rate, or at least improve one without worsening the other. It would also be useful to know which variables, if any, cause both metrics to worsen. In the following sub-sections, variables are analyzed for their effect on Training Time and Win Rate based upon their role in the experiment. Analysis below is separated based on whether the variable is an input to a DRL algorithm, a characteristic of the environment, or other. "Combined" and "overall" analysis refer to models built without regard to DRL algorithm.

1. Algorithm-Related Variables

Learning Rate and NN Structure demonstrated desirable behavior with respect to Training Time. Overall, increasing Learning Rate (-6.642, p=0.0773) and/or NN Structure (-21.836, p<0.0001) significantly reduced Training Time. When evaluated on a by-algorithm basis, increasing Learning Rate significantly reduced Training Time under PPO (-15.09, p=0.0184), but had no effect under VPG or TRPO. Increased NN Structure significantly reduced Training Time under PPO (-26.777, p=0.0001) and VPG (-35.612, p<0.0001), but had no effect under TRPO.

Overall, neither Learning Rate nor NN Structure had a significant effect on Win Rate. However, when analyzed by-algorithm, increases in Learning Rate had a large negative effect on Win Rate under VPG (-0.239, p=0.0997), while NN Structure had a large positive effect on Win Rate under PPO (0.584, p=0.0042). Therefore, it is generally advisable to increase both of these variables during training – subject to the previous caveat on NN Structure – but changes to Learning Rate may require extra consideration when using VPG.

When the data was combined, Changing Combat Model from Deterministic to Stochastic had a significant effect on Training Time under PPO (11.999, p=0.0619). Changing Combat Model had an overall negative effect (-0.036, p=0.0424) on Win Rate but extremely large positive effects under PPO

(0.477, p=0.0265) and VPG (0.306, p=0.0377). These results indicate that a mixed approach to Combat Model may be appropriate, i.e., training agents using a Deterministic model while employing them in an environment where a Stochastic model is used to adjudicate combat. For PPO, this was also accompanied by a marginal increase in training time (11.999, p=0.0619) but should still be considered as a potential tool for improving outcomes.

Training Duration is the most significant driver behind training time in all cases (Combined: 375.726, p<0.0001, PPO: 358.36, p<0.0001, VPG: 352.081, p<0.0001, TRPO: 398.524, p<0.0001). The effect of Training Duration on Win Rate was mixed. Win Rate increased under TRPO (0.075, p=0.00019) when Training Duration was increased but decreased at a substantially larger rate under PPO (-0.203, p=0.0329) and VPG (-0.171, p=0.0097). Generally, Training Duration can be decreased to reduce training time with little negative effect on Win Rate. However, under TRPO, Training Duration can be increased to increase Win Rate at the expense of higher Training Time.

Learning Factor and Delta had no effect on Training Time or Win Rate, whether looking at the combined data or on a by-algorithm basis. Therefore, these variables can be set to a convenient value.

2. Environment-Related Variables

In all cases, increasing Map Width resulted in increased Training Time (Combined: 19.308, p<0.0001, PPO: 27.353, p<0.0001, VPG: 17.13, p<0.0001, TRPO: 16.244, p<0.0001). As this experimented pitted two teams of opposing agents on a featureless battlefield, this result is reasonable. Excess Map Width presents opportunities for agents to wander without engaging their opponents. Reducing excess space on the map should reduce this wandering so that agents will require less Training Time.

The effect of increasing Map Width on Win Rate was more complicated. With combined data, this reduced (-0.065, p=0.0001) the Win Rate. By-algorithm, increasing Map Width increased the Win Rate under PPO (0.44, p=0.0412) but had no significant effect under VPG and TRPO. Based on these results, the existence of a relationship between Map Width and Win Rate cannot be conclusively determined and may warrant further testing.

Red Team Size requires a tradeoff. Increasing the number of red team agents tends to increase Win Rate (Combined: 0.04, p=0.0021, TRPO: 0.114, p=0.0209) but also requires more Training Time (Combined: 18.797, p<0.0001, TRPO 36.193, p<0.0001). Red Team Size had no effect on Training Time or Win Rate under PPO or VPG. Generally, we can expect a decrease in Training Time by decreasing the Red Team size, but at a cost of a lower Win Rate. Conversely, increasing Red Team Size should lead to higher Win Rate at the expense of higher Training Time.

When combined, increasing Blue Team Size increased Training Time (Blue Team Size: 73.106, p<0.0001) and decreased Win Rate (Blue Team Size: -0.3222, p<0.0001). Similar effect sizes and significance were observed for Training Time on a by-algorithm basis. However, Blue Team Size only had an algorithm-specific significant effect on Win Rate when using TRPO (-0.386, p<0.0001). The conclusion is that Blue Team Size should be reduced in order to reduce Training Time and increase Win Rate.

3. Other

Blue Alpha only affected Win Rate, and then only when using the PPO algorithm. This effect was quite large (-0.727, p=0.0008) though and so can't be easily overlooked. Overall and under other algorithms, changes in Blue Alpha had no effect on Training Time or Win Rate. However, if an AI red team is being trained to address a specific blue team strategy, then it may be worthwhile to decrease Blue Alpha in order to maximize the subsequent Win Rate. If the goal is to develop a generalizable red team though, then it may be necessary to accept the reduction in Win Rate that (potentially) accompanies an increase in Blue Alpha.

4. Training Time and Win Rate

With the exception of PPO, the effect of Training Time on Win Rate was not significant and even under PPO, was so small as to be negligible. This is a surprising result as an agent begins with no knowledge of how to defeat an enemy and should reasonably be expected to gain that knowledge as the time spent training increases. However, we present two possible explanations for this apparent contradiction.

The first is that additional Training Time does not equal additional time in combat with enemy agents. As previously mentioned, with no communications between the red team agents as to the location of any blue team agent, there was significant opportunity for the red team agents to individually wander in search of enemy. Even if the red team manages to defeat the blue team and increase its win count by one, time spend wandering during training would disproportionally increase its Training Time.

The second possible explanation is that the red team may have been overtrained. Given the particular variable settings of an individual match, there is a point beyond which additional training will not significantly increase the red team's ability to win. While this is similar to the above explanation in that Training Time increases with no corresponding increase in Win Rate, the cause is a matter of failing to end training once learning plateaus as opposed to agents wandering about unable to find an opponent to fight.

Unfortunately, data was not collected during the experiments that would determine the exact cause of, or potential solutions for, the lack of expected relationship between Training Time and Win rate. Further research is suggested to either refute these results or if confirmed, determine causes and solutions.

V. SUMMARY

In this work, we sought to determine how changes in scale and other training-specific variables might affect time required to train AI agents and their ability to win in combat, as measured by Training Time and Win Rate respectively, within a simulated wargame environment. A three-stage framework was developed in which a red team of agents learned to defeat a pre-trained blue team of agents and were then evaluated on how well they were able to overcome subsequent changes to the blue team's strategy.

Here we summarize the findings with respect to the research objectives first listed in section II OBJECTIVES.

- Is there a significant, quantifiable difference in the various reinforcement learning algorithms' abilities to train adversaries capable of defeating friendly forces?

  We found that while there was no difference in mean Training Time between the algorithms examined in this experiment, TRPO produced significantly higher Win Rates on average than PPO or VPG. This confirms that the choice of training algorithm can be a significant factor in the AI red team agent's ability to win in subsequent matches, particularly against a blue team whose strategy may be changing.

- Is there a quantifiable tradeoff between a reinforcement learning algorithm's ability to train adversaries capable of defeating friendly forces, and the time/resources required to train the adversary?

  We were unable to find any significant relationship between the red team's Training Time and the red team's Win Rate in subsequent matches against a blue team with an altered strategy. However, we have outlined some potential reasons for this phenomenon and suggest additional research.

- Given an adversary that has been reinforcement learning-trained to defeat friendly forces, how sensitive are the adversary's capabilities to subsequent changes in friendly force strategy and/or simulation environment?

  Generally, the red team's ability to win matches was not affected by changes in the blue team's strategy. Changes in blue team strategy negatively affected the red team's ability to win only when the red team was trained using the PPO algorithm. This effect was quite large though and so can't be easily overlooked.

  Training the red team using a deterministic model for adjudicating combat also tends to significantly increase the Win Rate in subsequent matches, whether those matches are adjudicated with a deterministic or stochastic model, with little effect on Training Time. The red team's ability to win in subsequent matches was also positively affected by a larger Red Team Size and negatively affected by a larger Blue Team Size or Map Width.

  Overall, we found the AI red team to be fairly resistant to changes in blue team strategy but highly sensitive to changes in the simulation environment.

- How do the answers to the above questions affect the ability to scale the simulations?

  All of the environment-specific variables considered would have a significant effect on the time required to train the red team and the ability of the red team to win against a changing blue team strategy if the scale of the simulation were changed. Increasing the Red Team Size tends to lengthen Training Time but improves the Win Rate. Increasing the Blue Team Size also increases Training Time but leads to a lower Win Rate. Likewise, increasing the size of the simulation map, Map Width, increases Training Time and lowers the red team's subsequent Win Rate.

  Although not simulation environment-related, we also found several algorithm-specific variables that had significant effects on the red team's Training

Time and subsequent Win Rate. Decreasing Training Duration, increasing Learning Rate and/or increasing the size of the NN Structure was observed to lead to decreased Training Time with minimal effect on Win Rate. Training the red team using a deterministic model for adjudicating combat also tends to increase the red team's Win Rate in subsequent matches, whether those matches are adjudicated with a deterministic or stochastic model, with little effect on the red team's Training Time. Some algorithm or experiment-specific variables, such as Blue Alpha, Learning Factor and Delta, did not have a significant effect on either Training Time or Win Rate.

This research was conducted on a simulated, open battlefield with identical agents on both teams seeking only to destroy the other team. Popular, off-the-shelf, well-supported DRL algorithms were selected for the experiment. Additionally, DRL neural networks can be much larger than the two-hidden layers of 96 nodes considered in this work, with much more complicated architectures connecting the nodes between layers. Therefore, this work should not be considered a definitive answer as to which variables can be changed to improve outcomes in all solutions, nor which algorithms are best, but rather a baseline from which to conduct further work in which environments and agents have more sophisticated characteristics and goals, alternative algorithms are employed, and/or the networks used to train the agents take on more sophisticated architectures.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

Table 12. Correlation Matrices

ALL										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	1.000	-0.041	0.041	0.012	0.041	0.040	0.002	-0.006	0.086	-0.159
Red Team Size	-0.041	1.000	-0.078	-0.001	-0.026	0.053	-0.012	-0.056	0.042	0.177
Blue Team Size	0.041	-0.078	1.000	0.001	-0.001	-0.026	0.012	0.125	0.200	-0.697
Training Duration	0.012	-0.001	0.001	1.000	0.027	-0.026	-0.012	-0.062	0.952	0.039
Learning Factor	0.041	-0.026	-0.001	0.027	1.000	-0.026	0.012	0.035	0.048	-0.008
Learning Rate	0.040	0.053	-0.026	-0.026	-0.026	1.000	0.040	-0.036	-0.040	0.028
Blue Alpha	0.002	-0.012	0.012	-0.012	0.012	0.040	1.000	-0.003	0.001	-0.010
Delta	-0.006	-0.056	0.125	-0.062	0.035	-0.036	-0.003	1.000	0.011	0.117
Training Time	0.086	0.042	0.200	0.952	0.048	-0.040	0.001	0.011	1.000	-0.080
Win Rate	-0.159	0.177	-0.697	0.039	-0.008	0.028	-0.010	0.117	-0.080	1.000
PPO										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	1.000	-0.046	0.087	0.042	0.046	0.000	0.004	▨	0.150	-0.255
Red Team Size	-0.046	1.000	-0.087	-0.046	-0.042	0.087	0.004	▨	-0.039	0.081
Blue Team Size	0.087	-0.087	1.000	0.000	0.000	-0.044	0.044	▨	0.226	-0.686
Training Duration	0.042	-0.046	0.000	1.000	0.046	0.000	0.004	▨	0.944	-0.058
Learning Factor	0.046	-0.042	0.000	0.046	1.000	0.000	-0.004	▨	0.081	-0.017
Learning Rate	0.000	0.087	-0.044	0.000	0.000	1.000	0.044	▨	-0.030	-0.020
Blue Alpha	0.004	0.004	0.044	0.004	-0.004	0.044	1.000	▨	0.051	0.041
Delta	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Training Time	0.150	-0.039	0.226	0.944	0.081	-0.030	0.051	▨	1.000	-0.209
Win Rate	-0.255	0.081	-0.686	-0.058	-0.017	-0.020	0.041	▨	-0.209	1.000
TRPO										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	1.000	-0.033	-0.031	-0.033	0.033	0.100	-0.001	0.001	0.012	-0.038
Red Team Size	-0.033	1.000	-0.067	0.067	0.000	0.000	-0.033	0.033	0.154	0.269
Blue Team Size	-0.031	-0.067	1.000	0.000	0.000	0.000	-0.031	0.031	0.154	-0.815
Training Duration	-0.033	0.067	0.000	1.000	0.000	-0.067	-0.033	0.033	0.973	0.147
Learning Factor	0.033	0.000	0.000	0.000	1.000	-0.067	0.033	0.033	0.020	-0.013
Learning Rate	0.100	0.000	0.000	-0.067	-0.067	1.000	0.033	-0.033	-0.087	0.028
Blue Alpha	-0.001	-0.033	-0.031	-0.033	0.033	0.033	1.000	-0.066	-0.060	-0.053
Delta	0.001	0.033	0.031	0.033	0.033	-0.033	-0.066	1.000	0.053	0.095
Training Time	0.012	0.154	0.154	0.973	0.020	-0.087	-0.060	0.053	1.000	0.025
Win Rate	-0.038	0.269	-0.815	0.147	-0.013	0.028	-0.053	0.095	0.025	1.000
VPG										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	1.000	-0.046	0.087	0.042	0.046	0.000	0.004	▨	0.128	-0.249
Red Team Size	-0.046	1.000	-0.087	-0.046	-0.042	0.087	0.004	▨	-0.032	0.172
Blue Team Size	0.087	-0.087	1.000	0.000	0.000	-0.044	0.044	▨	0.246	-0.557
Training Duration	0.042	-0.046	0.000	1.000	0.046	0.000	0.004	▨	0.942	-0.001
Learning Factor	0.046	-0.042	0.000	0.046	1.000	0.000	-0.004	▨	0.050	-0.006
Learning Rate	0.000	0.087	-0.044	0.000	0.000	1.000	0.044	▨	0.019	0.085
Blue Alpha	0.004	0.004	0.044	0.004	-0.004	0.044	1.000	▨	0.029	-0.040
Delta	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Training Time	0.128	-0.032	0.246	0.942	0.050	0.019	0.029	▨	1.000	-0.156
Win Rate	-0.249	0.172	-0.557	-0.001	-0.006	0.085	-0.040	▨	-0.156	1.000

Table 13. P-Values for Correlation Significant Tests

All										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	<.0001	0.481	0.481	0.833	0.481	0.493	0.978	0.950	0.136	0.005
Red Team Size	0.481	<.0001	0.173	0.990	0.656	0.360	0.833	0.553	0.469	0.002
Blue Team Size	0.481	0.173	<.0001	0.990	0.990	0.648	0.833	0.185	0.001	<.0001
Training Duration	0.833	0.990	0.990	<.0001	0.639	0.648	0.833	0.511	<.0001	0.496
Learning Factor	0.481	0.656	0.990	0.639	<.0001	0.648	0.833	0.710	0.409	0.889
Learning Rate	0.493	0.360	0.648	0.648	0.648	<.0001	0.493	0.704	0.493	0.627
Blue Alpha	0.978	0.833	0.833	0.833	0.833	0.493	<.0001	0.979	0.981	0.863
Delta	0.950	0.553	0.185	0.511	0.710	0.704	0.979	<.0001	0.909	0.213
Training Time	0.136	0.469	0.001	<.0001	0.409	0.493	0.981	0.909	<.0001	0.165
Win Rate	0.005	0.002	<.0001	0.496	0.889	0.627	0.863	0.213	0.165	<.0001
PPO										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	<.0001	0.667	0.409	0.693	0.667	1.000	0.971	▨	0.153	0.014
Red Team Size	0.667	<.0001	0.409	0.667	0.693	0.409	0.971	▨	0.710	0.446
Blue Team Size	0.409	0.409	<.0001	1.000	1.000	0.681	0.680	▨	0.031	<.0001
Training Duration	0.693	0.667	1.000	<.0001	0.667	1.000	0.971	▨	<.0001	0.584
Learning Factor	0.667	0.693	1.000	0.667	<.0001	1.000	0.971	▨	0.444	0.876
Learning Rate	1.000	0.409	0.681	1.000	1.000	<.0001	0.680	▨	0.775	0.853
Blue Alpha	0.971	0.971	0.680	0.971	0.971	0.680	<.0001	▨	0.632	0.699
Delta	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Training Time	0.153	0.710	0.031	<.0001	0.444	0.775	0.632	▨	<.0001	0.046
Win Rate	0.014	0.446	<.0001	0.584	0.876	0.853	0.699	▨	0.046	<.0001
TRPO										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	<.0001	0.718	0.735	0.718	0.718	0.277	0.990	0.990	0.901	0.680
Red Team Size	0.718	<.0001	0.468	0.469	1.000	1.000	0.718	0.718	0.093	0.003
Blue Team Size	0.735	0.468	<.0001	1.000	1.000	1.000	0.735	0.735	0.093	<.0001
Training Duration	0.718	0.469	1.000	<.0001	1.000	0.469	0.718	0.718	<.0001	0.111
Learning Factor	0.718	1.000	1.000	1.000	<.0001	0.469	0.718	0.718	0.827	0.892
Learning Rate	0.277	1.000	1.000	0.469	0.469	<.0001	0.718	0.718	0.344	0.758
Blue Alpha	0.990	0.718	0.735	0.718	0.718	0.718	<.0001	0.476	0.513	0.565
Delta	0.990	0.718	0.735	0.718	0.718	0.718	0.476	<.0001	0.566	0.304
Training Time	0.901	0.093	0.093	<.0001	0.827	0.344	0.513	0.566	<.0001	0.785
Win Rate	0.680	0.003	<.0001	0.111	0.892	0.758	0.565	0.304	0.785	<.0001
VPG										
	Map Width	Red Team Size	Blue Team Size	Training Duration	Learning Factor	Learning Rate	Blue Alpha	Delta	Training Time	Win Rate
Map Width	<.0001	0.667	0.409	0.693	0.667	1.000	0.971	▨	0.224	0.017
Red Team Size	0.667	<.0001	0.409	0.667	0.693	0.409	0.971	▨	0.760	0.102
Blue Team Size	0.409	0.409	<.0001	1.000	1.000	0.681	0.680	▨	0.018	<.0001
Training Duration	0.693	0.667	1.000	<.0001	0.667	1.000	0.971	▨	<.0001	0.995
Learning Factor	0.667	0.693	1.000	0.667	<.0001	1.000	0.971	▨	0.639	0.957
Learning Rate	1.000	0.409	0.681	1.000	1.000	<.0001	0.680	▨	0.859	0.423
Blue Alpha	0.971	0.971	0.680	0.971	0.971	0.680	<.0001	▨	0.785	0.706
Delta	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Training Time	0.224	0.760	0.018	<.0001	0.639	0.859	0.785	▨	<.0001	0.138
Win Rate	0.017	0.102	<.0001	0.995	0.957	0.423	0.706	▨	0.138	<.0001

Table 14. Training Time Full Models

	PPO			VPG			TRPO		
Term	Estimate	Std Error	Prob>|t|	Estimate	Std Error	Prob>|t|	Estimate	Std Error	Prob>|t|
Intercept	548.6348	5.545927	<.0001	535.8835	1.77972	<.0001	604.2748	2.423381	<.0001
Map Width	27.35314	6.038547	<.0001	17.13074	1.937804	<.0001	16.24479	2.8734	<.0001
Red Team Size	4.899031	6.469421	0.4521	2.497753	2.076074	0.2341	36.19321	2.709249	<.0001
Blue Team Size	75.93179	6.207401	<.0001	88.67028	1.991991	<.0001	62.77187	2.708236	<.0001
Training Duration	358.3601	6.452641	<.0001	352.081	2.07069	<.0001	398.5247	3.01018	<.0001
Learning Factor	8.483058	6.035747	0.1655	2.644132	1.936906	0.1778	1.286219	2.731853	0.6392
Learning Rate	-15.0902	6.208844	0.0184	-0.66188	1.992454	0.741	-1.26363	2.991701	0.674
NN Structure[(32,)]	-26.7777	6.412599	0.0001	-35.6128	2.05784	<.0001	-3.73703	2.685317	0.1682
Combat Model[Deterministic]	11.99971	6.295859	0.0619	-0.87162	2.020377	0.6679	2.658324	2.718474	0.3313
Map Width*Red Team Size	-5.40421	6.421072	0.4036	0.091665	2.060559	0.9647	-1.03166	2.731597	0.7068
Map Width*Blue Team Size	-3.2446	6.042021	0.5934	1.165569	1.938919	0.5502	0.024626	2.839157	0.9931
Map Width*Training Duration	20.68412	6.196748	0.0015	8.590916	1.988572	<.0001	13.8153	2.925329	<.0001
Map Width*Learning Factor	2.518827	6.733354	0.7098	0.496616	2.160772	0.8191	-1.35846	2.713347	0.6181
Map Width*Learning Rate	-9.93395	6.621622	0.1393	-10.8015	2.124917	<.0001	3.732275	2.8628	0.1964
Map Width*NN Structure[(32,)]	7.502948	6.210004	0.2321	4.596851	1.992826	0.0249	-2.08838	2.804146	0.4588
Map Width*Combat Model[Deterministic]	-2.20454	7.032468	0.7551	-7.36413	2.256759	0.0019	-4.92959	2.875065	0.0906
Red Team Size*Blue Team Size	-12.1037	6.212309	0.0565	-20.0027	1.993566	<.0001	-1.70504	2.940993	0.5638
Red Team Size*Training Duration	3.310198	6.734553	0.625	5.178245	2.161157	0.02	20.80863	2.610506	<.0001
Red Team Size*Learning Factor	-2.18472	6.629732	0.743	-0.79604	2.127519	0.7097	-0.6067	2.579504	0.8147
Red Team Size*Learning Rate	-7.80731	6.285777	0.2195	-1.80216	2.017142	0.3755	1.22402	2.703799	0.6521
Red Team Size*NN Structure[(32,)]	7.279009	6.352063	0.2568	5.585089	2.038414	0.0083	-3.1341	2.776934	0.2627
Red Team Size*Combat Model[Deterministic]	-1.09236	6.852069	0.8739	4.642191	2.198868	0.0393	1.736773	2.674099	0.518
Blue Team Size*Training Duration	52.27482	6.334985	<.0001	60.64408	2.032933	<.0001	42.16584	2.737776	<.0001
Blue Team Size*Learning Factor	0.259294	6.804479	0.9697	-3.3163	2.183597	0.1346	3.621512	2.707331	0.1851
Blue Team Size*Learning Rate	-6.40015	6.041812	0.2941	3.949734	1.938852	0.0465	-0.87854	2.763715	0.7515
Blue Team Size*NN Structure[(32,)]	-6.04938	6.849894	0.381	-9.25246	2.198171	<.0001	0.051518	2.70511	0.9849
Blue Team Size*Combat Model[Deterministic]	-1.5758	6.284491	0.8029	-1.71191	2.016729	0.3996	5.131846	2.697466	0.061
Training Duration*Learning Factor	15.03796	6.792483	0.031	1.939009	2.179747	0.3776	1.850441	2.705869	0.4962
Training Duration*Learning Rate	-9.32837	6.679837	0.1682	3.298389	2.143598	0.1296	-3.72915	3.018345	0.2206
Training Duration*NN Structure[(32,)]	-18.7598	6.060126	0.0031	-22.8196	1.944729	<.0001	1.568932	3.006856	0.6034
Training Duration*Combat Model[Deterministic]	13.8654	6.422075	0.0352	3.461028	2.060881	0.0987	1.799215	2.724545	0.5111
Learning Factor*Learning Rate	-2.32152	7.375998	0.7541	-5.3448	2.367	0.0279	-7.65803	2.908909	0.0103
Learning Factor*NN Structure[(32,)]	-6.76477	6.404735	0.2955	-0.72207	2.055316	0.7267	2.324519	2.863308	0.4195
Learning Factor*Combat Model[Deterministic]	1.360818	6.810728	0.8424	-5.22316	2.185602	0.0203	4.047957	2.802279	0.1528
Learning Rate*NN Structure[(32,)]	2.264058	6.066338	0.7104	-9.17811	1.946723	<.0001	-1.64697	3.202701	0.6086
Learning Rate*Combat Model	6.043313	7.053572	0.3953	6.090063	2.263532	0.0094	-4.74998	2.844956	0.0992
NN Structure[(32,)]*Combat Model[Deterministic]	-6.1425	6.810749	0.3711	-1.43182	2.185608	0.5151	0.536576	2.934393	0.8554
Delta							-4.76304	2.868742	0.1011
Map Width*Delta							-0.14907	2.92661	0.9595
Red Team Size*Delta							-5.30897	2.662556	0.0498
Blue Team Size*Delta							-0.07895	2.735208	0.9771
Training Duration*Delta							2.443057	2.736432	0.3749
Learning Factor*Delta							1.888187	2.65155	0.4786
Learning Rate*Delta							0.966764	2.687959	0.7201
Delta*NN Structure[(32,)]							-4.85847	2.694638	0.0755
Delta*Combat Model[Deterministic]							-2.57005	2.747315	0.3526

Table 15. Win Rate Full Models

	PPO			VPG			TRPO		
Term	Estimate	Std Error	Prob>|t|	Estimate	Std Error	Prob>|t|	Estimate	Std Error	Prob>|t|
Intercept	-0.18185	0.220112	0.413	-0.17372	0.151377	0.2571	0.378161	0.017411	<.0001
Map Width	0.440362	0.209687	0.0412	0.104951	0.144208	0.4704	-0.03416	0.022415	0.1325
Red Team Size	0.14089	0.216302	0.5181	0.191228	0.148757	0.2051	0.114016	0.020959	<.0001
Blue Team Size	0.01714	0.205085	0.9338	-0.19424	0.141043	0.1751	-0.38685	0.021274	<.0001
Training Duration	-0.20358	0.092561	0.0329	-0.17182	0.063657	0.0097	0.075317	0.023189	0.0019
Learning Factor	-0.27369	0.209748	0.1984	0.061167	0.14425	0.6735	0.011119	0.020593	0.5911
Learning Rate	0.082023	0.207489	0.6944	-0.23979	0.142696	0.0997	0.020739	0.022275	0.3553
Blue Alpha	-0.7278	0.203044	0.0008	0.048448	0.139639	0.7302	-0.01531	0.020941	0.4673
NN Structure[(32,)]	0.584996	0.194276	0.0042	0.076377	0.133609	0.5703	-0.03007	0.020713	0.1514
Combat Model[Deterministic]	0.477403	0.208236	0.0265	0.306491	0.14321	0.0377	-0.03281	0.022844	0.1558
Map Width*Red Team Size	0.018411	0.034341	0.5945	-0.0294	0.023618	0.2195	0.001391	0.022538	0.951
Map Width*Blue Team Size	0.103232	0.032588	0.0027	0.131454	0.022412	<.0001	0.043654	0.023333	0.0659
Map Width*Training Duration	0.223569	0.084661	0.0113	0.072715	0.058224	0.218	0.094036	0.023039	0.0001
Map Width*Learning Factor	0.037652	0.038231	0.3299	-0.01888	0.026293	0.4764	-0.01656	0.021731	0.449
Map Width*Learning Rate	-0.00971	0.03852	0.8022	0.157698	0.026492	<.0001	0.055175	0.02328	0.0208
Map Width*Blue Alpha	0.008389	0.036106	0.8173	-0.0374	0.024831	0.1389	0.007708	0.025192	0.7606
Map Width*NN Structure[(32,)]	0.103558	0.033599	0.0035	-0.01349	0.023107	0.5622	-0.00135	0.020971	0.9491
Map Width*Combat Model[Deterministic]	-0.097	0.039448	0.0178	0.030852	0.02713	0.2613	-0.04532	0.0226	0.0492
Red Team Size*Blue Team Size	-0.08434	0.0339	0.0165	-0.02753	0.023314	0.2437	-0.05795	0.022621	0.0128
Red Team Size*Training Duration	0.076576	0.090922	0.404	0.048792	0.06253	0.4392	0.017937	0.022486	0.428
Red Team Size*Learning Factor	-0.04092	0.03479	0.2455	0.024808	0.023926	0.3052	-0.02952	0.020903	0.1628
Red Team Size*Learning Rate	-0.03404	0.034198	0.3248	-0.0511	0.023519	0.035	0.053072	0.02141	0.0158
Red Team Size*Blue Alpha	-0.05309	0.034014	0.1254	0.014709	0.023392	0.5326	-0.03683	0.023625	0.124
Red Team Size*NN Structure[(32,)]	0.024776	0.035582	0.4897	0.127293	0.024471	<.0001	0.01587	0.021991	0.4731
Red Team Size*Combat Model[Deterministic]	0.054574	0.038603	0.1642	0.03375	0.026548	0.21	-0.01107	0.020591	0.5926
Blue Team Size*Training Duration	0.14567	0.083537	0.0879	0.028826	0.057451	0.6182	-0.05446	0.021495	0.0137
Blue Team Size*Learning Factor	0.000103	0.038307	0.9979	0.009858	0.026345	0.71	-0.02442	0.020346	0.2345
Blue Team Size*Learning Rate	0.043232	0.032196	0.1859	-0.02848	0.022142	0.2047	0.004736	0.020483	0.8179
Blue Team Size*Blue Alpha	-0.0121	0.037053	0.7456	0.039903	0.025483	0.1242	0.007799	0.021784	0.7215
Blue Team Size*NN Structure[(32,)]	0.011092	0.035783	0.758	-0.06509	0.024609	0.0111	0.011997	0.020075	0.5522
Blue Team Size*Combat Model[Deterministic]	0.030911	0.033389	0.3594	0.003507	0.022963	0.8793	0.010459	0.020187	0.6062
Training Duration*Learning Factor	-0.11864	0.08745	0.1815	0.047184	0.060142	0.4367	-0.02956	0.021076	0.1656
Training Duration*Learning Rate	0.059995	0.089333	0.5052	-0.11271	0.061437	0.073	-0.02602	0.023857	0.2795
Training Duration*Blue Alpha	-0.29796	0.084129	0.0009	0.030995	0.057858	0.5947	0.018489	0.023025	0.425
Training Duration*NN Structure[(32,)]	0.226641	0.079577	0.0066	-0.00372	0.054727	0.9462	0.052431	0.022345	0.0221
Training Duration*Combat Model[Deterministic]	0.192259	0.087181	0.0325	0.159777	0.059957	0.0106	0.057971	0.020314	0.0058
Learning Factor*Learning Rate	-0.022	0.038836	0.5739	-0.06579	0.026709	0.0176	-0.0633	0.023435	0.0088
Learning Factor*Blue Alpha	0.030898	0.03287	0.3521	0.072939	0.022606	0.0023	0.004734	0.021889	0.8295
Learning Factor*NN Structure[(32,)]	-0.0309	0.035614	0.3901	0.026675	0.024493	0.2818	0.00555	0.021332	0.7956
Learning Factor*Combat Model[Deterministic]	0.10891	0.035913	0.004	0.064146	0.024698	0.0126	0.025529	0.021065	0.23
Learning Rate*Blue Alpha	0.025575	0.034892	0.4673	-0.03664	0.023996	0.1336	0.016592	0.021457	0.4422
Learning Rate*NN Structure[(32,)]	-0.01713	0.032695	0.6029	0.082273	0.022485	0.0007	0.036128	0.024448	0.1444
Learning Rate*Combat Model[Deterministic]	0.118453	0.041899	0.0069	-0.07536	0.028815	0.012	0.019567	0.02089	0.3525
Blue Alpha*NN Structure[(32,)]	0.027043	0.032801	0.4139	-0.0084	0.022559	0.7114	0.017357	0.018416	0.3495
Blue Alpha*Combat Model[Deterministic]	-0.06454	0.034391	0.0669	0.001891	0.023652	0.9366	-0.00035	0.019831	0.9859
NN Structure*Combat Model[Deterministic]	0.08405	0.03581	0.0233	0.015697	0.024628	0.527	-0.03171	0.022076	0.1558
Delta							0.020586	0.021736	0.3472
Map Width*Delta							0.018033	0.021587	0.4066
Red Team Size*Delta							0.03352	0.022058	0.1335
Blue Team Size*Delta							-0.04153	0.022056	0.0643
Training Duration*Delta							0.002567	0.020707	0.9017
Learning Factor*Delta							-0.03006	0.019951	0.1368
Learning Rate*Delta							0.009657	0.020164	0.6336
Blue Alpha*Delta							-0.01192	0.02038	0.5606
Delta*NN Structure[(32,)]							-0.02173	0.019845	0.2777
Delta*Combat Model[Deterministic]							0.01796	0.020908	0.3936

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Anaconda, Retrieved January 3, 2021, from https://www.anaconda.com/products/individual

Anthony, T., Tian, Z., Barber, D. (2017). Citation: Thinking fast and slow with deep learning and tree search. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, pp. 5366-5376.

Berner, C. Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C. Farhi, et al. (2019). *Dota 2 with large scale deep reinforcement learning*. arXiv preprint arXiv:1912.06680

Boron, J. (2020). *Developing Combat Behavior Through Reinforcement Learning* (MS thesis). Available from Calhoun database, https://calhoun.nps.edu/handle/10945/65414

Boron, J., Darken, C. (2020). *Developing Combat Behavior through Reinforcement Learning in Wargames and Simulations*. 2020 IEEE Conference on Games (CoG), Osaka, Japan, pp. 728-731, https://doi.org/10.1109/CoG47356.2020.9231609

Goodman, J., Sebastian, R., Lucas, S. (2020). *AI and Wargaming*. arXiv: 2009.08922 [cs.AI]

Gym, Retrieved January 3, 2021, from https://gym.openai.com/docs/

Hansen, N. Ostermeier, A. (2001). *Completely Derandomized Self-Adaptation in Evolution Strategies*. Evol Comput 1, 9 (2), pp. 159-195, https://doi.org/10.1162/106365601750190398

Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., et al. (2020). *Acme: A Research Framework for Distributed Reinforcement Learning*. arXiv: 2006.00979 [cs.LG]

JMP, Retrieved January 3, 2021, from https://www.jmp.com/en_us/home.html

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., et al. (2018). *Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation*. Proceedings of The 2nd Conference on Robot Learning, vol. 87, pp. 651-673

Khadka, S., Tumer, K. (2018). *Evolution-Guided Policy Gradient in Reinforcement Learning*. Proceedings of the 32nd International Conference on

Neural Information Processing Systems, pp. 1196-1208, https://doi.org/10.5555/3326943.3327053

Lucek, S., Collander-Brown, S. (2017). *Using Artificial Intelligence Algorithms for High Level Tactical Wargames and New Approaches to Wargame Simulation*. Journal of Applied Operational Research, vol. 9, no. 1, pp. 11-26

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. et al. (2013). *Playing Atari with Deep Reinforcement Learning*. arXiv: 1312.5602 [cs.LG]

Mnih, V., Kavukcuoglu, V., Silver, C., Rusu, A., Veness, J., Bellemare, M., et al. (2015). *Human-level control through deep reinforcement learning*. Nature, vol. 518, issue 7540, pp. 529-533

Moy G., Shekh S. (2019) *The Application of AlphaZero to Wargaming*. AI 2019: Advances in Artificial Intelligence. AI 2019. Lecture Notes in Computer Science, vol 11919. Springer. https://doi.org/10.1007/978-3-030-35288-2_1

Muñoz, J., Gutierrez, G., Sanchis, A. (2012). *Towards imitation of human driving style in car racing games*. Believable Bots, pp. 289-313, Springer

Ng, A., Russell, S. (2000). *Algorithms for Inverse Reinforcement Learning*. Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00). pp. 663-670, Morgan Kaufmann Publishers Inc.

Spinning Up, Retrieved January 3, 2021, from https://spinningup.openai.com/en/latest/

Spinning Up, *Why These Algorithms*? Retrieved January 3, 2021, from https://spinningup.openai.com/en/latest/user/algorithms.html#why-these-algorithms

Pentreath, N. (June 22-26, 2020). *Scaling up Deep Learning by Scaling Down*. SPARK+AI Summit 2020, virtual event, url: https://databricks.com/session_na20/scaling-up-deep-learning-by-scaling-down

Perez, D., Samothrakis, S., Lucas, S., Rohlfshagen, P. (2013). *Rolling horizon evolution versus tree search for navigation in single-player real-time games*. Proceedings of the 15th annual conference on Genetic and evolutionary computation (GECCO '13). Association for Computing Machinery, pp. 351-358. https://doi.org/10.1145/2463372.2463413

Perez-Liebana, D., Gaina, R. D., Drageset, O., İlhan, E., Balla, M., & Lucas, S. M. (2019). *Analysis of Statistical Forward Planning Methods in Pommerman*.

Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 15(1), pp. 66-72.

Pournelle, P. (2017). *Designing Wargames for the Analytic Purpose*. Phalanx, vol. 50, no. 2, pp. 48-53. JSTOR

Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I. (2017). *Evolution Strategies as a Scalable Alternative to Reinforcement Learning*. arXiv: 1703.03864 [stat.ML]

Schwartz, P., O'Neill, D., Bentz, M., Brown, A., Doyle, B., Liepa, O., et al. (2020). *AI-enabled Wargaming in the Military Decision Making Process*. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, vol. 11413, pp. 118-134, https://doi.org/10.1117/12.2560494

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. Science, vol. 362, nbr. 6419, pp. 1140-1144, https://doi.org/10.1126/science.aar6404

Stanescu, M., Barriga, N., Hess, A., Buro, M. (2016). *Evaluating real-time strategy game states using convolutional neural networks*. 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1-7, https://doi.org/10.1109/CIG.2016.7860439

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). *Grandmaster level in StarCraft II using multi-agent reinforcement learning*. Nature, vol. 575, issue 7782, pp. 350-354.

Wade, B. (2018). *The Four Critical Elements of Analytic Wargame Design*. Phalanx, vol. 51, no. 4, pp. 18-23. JSTOR

Wang, H., Tang, H., Hao, J., Hao, X., Fu, Y., Ma, Y. (2020). *Large Scale Deep Reinforcement Learning in War-games*. 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Seoul, Korea (South), pp. 1693-1699. https://doi.org/10.1109/BIBM49941.2020.9313387

Watkins, C. (1989). "Learning from Delayed Rewards," (Ph.D. thesis), Cambridge University

Zhang, J., Xue, Q. (2020). *Actor–critic-based decision-making method for the artificial intelligence commander in tactical wargames*. The Journal of Defense Modeling and Simulation. https://doi.org/10.1177/1548512920954542

Zhi, J., Wang, R., Clune, J., Stanley, K. (June 30, 2020), *Fiber: Distributed Computing for AI Made Simple*. Retrieved January 3, 2021, from https://eng.uber.com/fiberdistributed/