On games and simulators as a platform for development of artificial intelligence for command and control

Vinicius G Goecks¹, Nicholas Waytowich², Derrik E Asher², Song Jun Park², Mark Mittrick², John Richardson², Manuel Vindiola², Anne Logie², Mark Dennison³, Theron Trout², Priya Narayanan², and Alexander Kott²

Abstract
Games and simulators can be a valuable platform to execute complex multi-agent, multiplayer, imperfect information scenarios with significant parallels to military applications: multiple participants manage resources and make decisions that command assets to secure specific areas of a map or neutralize opposing forces. These characteristics have attracted the artificial intelligence (AI) community by supporting development of algorithms with complex benchmarks and the capability to rapidly iterate over new ideas. The success of AI algorithms in real-time strategy games such as StarCraft II has also attracted the attention of the military research community aiming to explore similar techniques in military counter-part scenarios. Aiming to bridge the connection between games and military applications, this work discusses past and current efforts on how games and simulators, together with the AI algorithms, have been adapted to simulate certain aspects of military missions and how they might impact the future battlefield. This paper also investigates how advances in virtual reality and visual augmentation systems open new possibilities in human interfaces with gaming platforms and their military parallels.

Keywords
Artificial intelligence, reinforcement learning, war gaming, command and control, human–computer interface, future battlefield

1. Introduction
In warfare, the ability to anticipate the consequences of the opponent’s possible actions and counteractions is an essential skill.¹ This becomes even more challenging in the future battlefield and multi-domain operations where the speed and complexity of operations are likely to exceed the cognitive abilities of a human command staff in charge of conventional, largely manual, command, and control C2 processes. Strategy games such as chess, poker, and Go have abstracted some of C2 concepts. While wargames are often manual games played with physical maps, markers, and spreadsheets to compute battle outcomes,² computer games and modern game engines such as Unreal Engine (https://www.unrealengine.com/) and Unity (https://unity.com/) are capable of automating simulation of complex battle and related physics simulations.

Computer games and game engines are not only suitable to simulate military scenarios but also provide a useful test-bed for developing artificial intelligence (AI) algorithms. Games such as StarCraft II,³⁴⁹ Dota 2,¹⁰ Atari,¹¹⁻¹³ Go,¹⁴⁻¹⁶ chess,¹⁷,¹⁸ and heads-up no-limit poker¹⁹ have all been used as platforms for training
artificially intelligent agents. Another capability developed by the gaming industry that is relevant to the military domain is virtual reality and visual augmentation systems, which may potentially provide commanders and staff a more intuitive and information-rich display of the battlefield.

Recently, there has been a major effort toward leveraging the capabilities of computer games and simulation engines to support military C2 planning due to their suitability for integrating AI algorithms, some examples of which we discuss shortly. The work described in this paper focuses on adapting imperfect information real-time strategy (RTS) games and their corresponding simulators, to emulate military assets in battlefield scenarios. Furthermore, this work aims to advance the research and development efforts associated with AI algorithms for C2. Flexibility in these games and game engines allow for simulations of a rather broad range of terrains or assets. In addition, most games and simulators allow for battlefield scenarios to be played out at a “faster then real-time” speed which is a highly desirable feature for the rapid development and testing of data-driven algorithms for military applications.

The paper is organized as follows. In section “Background and related work,” we offer motivation for research on AI-based tools for C2 decision-making support, and describe a number of prior-related efforts to develop approaches to such tools. In particular, we offer arguments in support of using machine learning that can leverage data produced using simulation/wargaming runs, to achieve affordable AI solutions. Then, we describe two case studies. In the first, we describe how we adapted a fast and popular gaming system to approximate partly realistic military operation and how we used it in conjunction with a reinforcement learning (RL) algorithm to train an “artificial commander” (a trained agent) that commands a blue force (BLUFOR) to fight an opposing enemy force (red force (OPFOR)). In the second case study, we describe a similar exploration using a realistic military simulation system. Then, we discuss approaches to overcoming common challenges that are likely to arise in transitioning such future technologies to real-world battlefield, and also describe our key findings.

The key contributions of the research described in the paper are two-fold: first, we empirically demonstrate that deep RL algorithms can significantly outperform both human and doctrine-based baselines without using any pre-coded expert knowledge, and learn effective behaviors entirely from experience. Second, we formulate and empirically confirm a set of important aspects and requirements of developing a training system for AI algorithms in military-relevant games and battlefield simulators for C2. In effect, these are initial recommendations for the researchers who build related experimental and developmental systems.

2. Background and related work

We start by defining important terms used throughout this research work. War game is defined in this work as a largely manual, strategy game that uses rules, data, and procedures to simulate an armed conflict between two or more opposing sides, which is used to train military officers and plan military courses of action (COAs). This is different from games, which is used here as a fully automated, recreational computer application with well-defined rules and score systems that uses a simulation of an armed conflict as a form of entertainment. Similarly, simulators are used here as a hybrid form of wargames and games. Simulators are fully automated computer applications that aim to realistically simulate the outcome of military battles and COAs. They are not designed as an entertainment platform but as a tool to aid military planning.

Concerning C2, we use the same definition as command and control war fighting function defined in the Army Doctrine Publication No. 3—Operations as the “related tasks and a system that enable commanders to synchronize and converge all elements of combat power.” Similarly, AI-support to C2 approaches are AI systems developed to aid human commanders by providing additional information or recommendations in the C2 process.

The past few decades have seen a number of ideas and corresponding research toward developing automated or semi-automated tools that could support decision-making in planning and executing military operations. Defense Advanced Research Projects Agency (DARPA’s) JFACC program took place in the late 1990s and developed a number concepts and prototypes for agile management of a joint air battle. Most of the approaches considered at that time involved continuous real-time optimization and re-optimization (as situation continually changes) of routes and activities of various air assets.

Also in the mid-to-late 1990s, the Army funded the CADET project which explored classical hierarchical planning, adapted for adversarial environments, for transforming high-level battle sketch into a detailed synchronization matrix—a key product of the doctrinal Military Decision-Making Process (MDMP).

In the early 2000s, DARPA initiated the RAID project which explored a number of technologies for anticipating enemy battle plans, as well as dynamically proposing friendly tactical actions. At the time, game-solving algorithms emerged as the most successful among the technological approaches explored.

The role of multiple domains and their extremely complex interactions—beyond the traditional kinetic fights to
include political, economic, and social effects—were explored in the late 2000s in DARPA’s COMPOEX program. This program investigated the use of interconnected simulation sub-models, mostly system-dynamic models, in order to assist senior military and civilian leaders in planning and executing large-scale campaigns in complex operational environments. The importance of non-traditional warfighting domains such as the cyber domain has been recognized and studied in the mid 2010s by an NATO research group that looked into simulation approaches to assessing mission impacts of cyberattacks and highlighted strong non-linear effects of interactions between cyber, human, and traditional physical domains.

All approaches taken in the research efforts mentioned above—and many other similar ones—have major and somewhat common weaknesses. They tend to require a rigid, precise formulation of the problem domain. Once such a formulation is constructed, they tend to produce effective results. However, when a new element needs to be incorporated into the formulation (e.g., a new type of a military asset or a new tactic), a difficult, expensive, manual, and long effort is required to “rewire” the problem formulation and to fine-tune the solution mechanism. And the real world endlessly presents a stream of new elements and entities that must be taken into account. In rule-based systems of the 1980s, a system would become unmanageable as more and more rules (with often unpredictable interactions) had to be added to represent the real-world intricacies of a domain. Similarly, in optimization-based approaches, an endless number of relations between significant variables, and variety of constraints had to be continually and manually be added (a maintenance nightmare) to represent real-world intricacies of a domain. In game-based approaches, the rules governing legal moves and effects of moves for each piece would gradually become hopelessly convoluted as more and more realities of a domain had to be manually contrived and added to game formulation.

In short, AI-support to C2 approaches are costly in their representation-building and maintenance. Ideally, we would like to see a system that learns its problem formulation and solution algorithm directly from its experiences in a real or simulated world, without any (or with little) manual programming. Machine learning, particularly RL offers that promise.

Walsh et al. investigated how the military can use the capacity of dealing with large volumes of data and greater decision speed of machine learning algorithms for military planning and C2. They analyzed and rated the characteristics of 10 games, such as Go, Bridge, and StarCraft II, and 10 C2 processes, such as intelligence preparation of the battlefield, operational assessment, and troop leading procedures. Example of characteristics is the operational tempo, rate of environment change, problem complexity, data availability, stochasticity of action outcomes, and others. Their main conclusion related to using games as a platform for military applications was that real-world tasks are very different from many of the games and environments used to develop and demonstrate AI systems, which is mostly due to them having fixed and well-defined rules that are regularly exploited by AI agents.

Related to these games, StarCraft II (SC2) is an RTS game where players compete for map domination and resources. The players need to manage resources to expand their bases, build more units, upgrade them, and coordinate tens of units at the same time to defeat their opponent. Vinyals et al. presented AlphaStar, the first AI agent to learn how to play the full game of SC2 at the highest competitive level. AlphaStar played under the same kind of constraints that humans play in professionally approved conditions. The AI agent was able to achieve this success using a combination of self-play via RL, multi-agent learning, and imitation learning using hundreds of thousands of expert replays. Sun et al. proposed the TStarBot AI agent, a combination of RL and encoded knowledge of the game, becoming the first learning agent to defeat all levels of the built-in rule-based AI in the StarCraft II game. Han et al. extended that to TStarBot-X with a complete analysis of the learned agent trained under a limited scale of computation resources. Also using limited computational resources, Wang et al. presented StarCraft Commander, an AI system that also achieves the highest competitive level but using a model with a third of the parameters used by AlphaStar and about one-sixth of the training time.

OpenAI et al. was the first AI system to defeat the world champions at the Dota 2 Esport game, a partially observable five-against-five RTS game with high dimensionality of observation and action spaces. In the Dota 2 game, both teams compete for control of strategic locations of the map and to defend their bases at the corners of the map. Each agent controls a special hero unit with unique abilities that are used when teams engage in conflict against each other and non-player-controlled units. OpenAI was able to solve this complex task by scaling existing RL algorithms, in this case, Proximal Policy Optimization (PPO) with Generalized Advantage Estimation (GAE), to train using thousand of graphics processing units (GPUs) over 10 months with a specialized neural-network architecture to handle the long time horizons of the game.

Boron and Darken investigated the use of deep reinforcement algorithms to solve simulated, perfect information, turn-based, multi-agent, and small tactical engagement scenarios based in military doctrine. The grid-world environment emulated an offensive scenario where the homogeneous units controlled by the RL agent could move and attack and the defender was static and combat

**References:**
1. Goecks et al.
2. Walsh et al. 28, 29
3. Vinyals et al.
4. Sun et al.
5. Han et al.
6. Wang et al.
7. Vinyals et al.
8. OpenAI et al.
was resolved using both deterministic and stochastic Lanchester combat models. One of the main results was how the learned behavior can be controlled to follow different principles of war such as mass or force based on the discount factor for the rewards.

Asher et al. proposed the adoption of militarily relevant tactics to a simulated predator–prey pursuit task that allowed for teams of deep RL agents to engage in various tactics through capability modifications such as decoys, traps, and camouflage. Furthermore, this research group has introduced methods for measuring coordination toward a framework for integrating AI agents into mixed soldier-agent teams.

With respect to simulators built using existing game engines, Fu et al. applied deep RL to the C2 of air defense operations. Digital battlefield environment based on Unreal Engine was developed for RL training process. The training setup consisted of static and random opponent strategies where attack route and formation of units were either fixed or random. For evaluation, win rate, battle damage statistics, and combat details were compared against human experts. Their experimental results showed that an agent trained using deep RL techniques achieved higher winning rate than the human experts in fixed and random opponent scenarios. Conversely, in this work, we are proposing a real-time decision-making and planning system that is designed using RL formalism, which can operate in tandem with the commander.

With respect to military simulators, the AlphaDogfight Trial was a DARPA sponsored competition where an AI controlled a simulated F-16 fighter jet in an aerial combat against an Air Force pilot flying in a virtual reality simulator. First place winner, Heron Systems, designed an F-16 AI, which outperformed seven other participating companies and defeated an experienced Air Force pilot with a score of 5–0 in a simulated dogfight. The outcome of this competition demonstrated that an AI-based pilot can provide precise aircraft maneuvering that may surpass human abilities. In addition, this F-16 AI opened possibilities of human–machine teaming such that human pilots can address high-level strategies while offloading low level, tedious tactical tasks to an AI-based agent.

Schwartz et al. described a wargaming support tool that used a genetic algorithm to recommend modifications to an existing friendly COA, a sequence of decisions to be taken in a military scenario, framed as task scheduling problem with multiple tasks and their respective start times. The system integrated user input to the optimization process to constrain which tasks can be modified, minimum and maximum start times, all the AI settings, and monitor all the recommendations that were being simulated by the AI. The authors showed that their system was able to generate expert-level recommendation to friendly COAs and support the MDMP.

Given these positive results on wargaming, real-time complex strategy and multi-agent recreational applications, and military simulators, extending these algorithms to military applications as a whole is a promising research avenue that we explore in this research work.

3. Games and simulations for deep RL of C2

In this work, we investigate whether deep RL algorithms might support future agile and adaptive C2 of multi-domain forces that would enable the commander and staff to exploit rapidly and effectively fleeting windows of superiority.

To train RL agents for a C2 scenario, a fast running simulator with the appropriate interfaces is needed to enable learning algorithms to run for millions of simulation steps, as is often required by state-of-the-art deep RL algorithms. Much of what motivated us to study AI in C2 applications was the difficulty of finding a simulator that had all the required features to develop machine learning algorithms for C2. These features include, but are not limited to, being able to:

- interact with the simulator, also known as the environment, via an application programming interface (API), which includes the ability to query environment states and send actions computed by the agent;
- simulate interactions between agent and environment faster than real-time;
- parallelize simulations for distributed training of the machine learning agent, ideally over multiple nodes;
- randomize environment conditions during training to learn more robust and generalizable RL policies; and
- emulate realistic combat characteristics such as terrain, mobility, visibility, communications, weapon range, damage, rate of fire, and other factors, in order to mimic real-world military scenarios.

While we were unable to find a simulator that met all these requirements out of the box, game-based simulators, such as the StarCraft II Learning Environment (SC2LE), were able to be adapted with the purpose of applying AI algorithms in potential military C2 applications. The next sections describe how we adapted SC2LE and the military simulator OpSim, respectively, to be used as our main
platform to develop an artificial commander for C2 tasks using RL.

3.1. Case study: StarCraft II for military applications

We developed a research prototype C2 simulation and experimentation capability that included simulated battle-spaces using the SC2LE with interfaces to deep RL algorithms via RLlib, a library that provides scalable software primitives for RL on a high-performance computing system.

StarCraft II has a number of difficult challenges for AI algorithms that make it a suitable simulation environment for wargaming, C2, and other military applications. For example, the game has complex state and action spaces, can last tens of thousands of time-steps, can have thousands of actions selected in real-time, and can capture uncertainty due to the partial observability or “fog-of-war.” Furthermore, the game has heterogeneous assets, an inherent control architecture that in some elements resembles military C2, embedded objectives that have adversarial nature, and a shallow learning curve for implementation/modification compared to more robust simulations.

DeepMind’s SC2LE framework exposes Blizzard Entertainment’s StarCraft II Machine Learning API as an RL environment. This tool provides access to StarCraft II, its associated map editor, and an interface for RL agents to interact with StarCraft II, getting observations and sending actions.

Figure 1. Illustration of areas of operations in the TigerClaw scenario.

Figure 2. StarCraft II using MILSTD2525 symbols.

Figure 3. Modified StarCraft II (left panel), and map to resemble real-world area of operation (right panel).

Using the StarCraft II Editor, we implemented TigerClaw, a brigade-scale offensive operation scenario to generate a tactical combat scenario within the StarCraft II simulation environment, as seen in Figure 1. The game was militarized by re-skinning the icons to incorporate military symbology, as seen in Figure 2.

In TigerClaw, the BLUFOR’s goal is to cross the wadi terrain, neutralize the Red Force, and control certain geographic locations. These objectives are encoded in the game score for use by RL agents as a baseline for comparison across different neural-network architectures and reward driving attributes. The following sections describe the process we used to adapt the map, units, and rewards to this military scenario.

3.1.1. Map development. We created a new Melee Map for TigerClaw scenario using the StarCraft II Editor. The map size was the largest available, 256 by 256 tiles, using the StarCraft II coordinate system. A wasteland tile set was used as the default surface of the map since it visually resembled a desert region in the area of operations in TigerClaw, as seen in Figure 3. After the initial setup, we
used the *Terrain* tools to modify the map to loosely approximate the area of operation. The key terrain feature was the impassable wadi with limited crossing points.

Distance scaling was an important factor for the scenario creation. In the initial map, we used the known distance between landmarks to translate *StarCraft II* distance, using its internal coordinate system, into kilometers and latitude and longitude. This translation is important for adjusting weapons range during unit modification and to ensure compatibility with other internal visualization tools that expect geographic coordinates as input.

### 3.1.2. Playable units modification.

To simulate the *TigerClaw* scenario, we selected *StarCraft II* “fantastic” units that could be made to approximate the capabilities of realistic military units, even if crudely. The *StarCraft II* units were first duplicated and their attributes modified in the Editor to support the scenario. First, we modified the appearance of the units and replaced it with an appropriate MIL-STD-2525C symbol, as seen in Table 1.

| TigerClaw unit       | StarCraft II unit       |
|----------------------|-------------------------|
| Armor                | Siege tank (tank mode)  |
| Mechanized infantry | Hellion                 |
| Mortar               | Marauder                |
| Aviation             | Banshee                 |
| Artillery            | Siege tank (siege mode) |
| Anti-armor           | Reaper                  |
| Infantry             | Marine                  |

Other attributes modified for the scenario included, weapon range, weapon damage, unit speed, and unit health (how much damage it can sustain). Weapon ranges were discerned from open-source materials and scaled to the map dimensions. Unit speed was established in the *TigerClaw* operations order and fixed at that value. The attributes for damage and health were estimated, with the guiding principle of maintaining a conflict that challenges both sides. Each *StarCraft II* unit usually had only one weapon making it challenging to simulate the variety of armaments available to a company size unit.

In addition, the BLUFOR units were modified so that they would not engage offensively or defensively unless specifically commanded by the player or learning agent in control. To control the OPFORs, we used two different strategies. The first strategy was to include a scripted COA, a set of high-level actions to take, for OPFOR movements that is executed in every simulation. The units default aggressiveness attributes controlled how it engaged Blue. The second strategy was to let a *StarCraft II* bot AI control the OPFOR to execute an all-out attack, or suicide as it is termed in the *Editor*. The built-in *StarCraft II* bot has several difficulty levels (1–10) which dictate the proficiency of the bot. The bot levels indicate their proficiency where a level 1 is a fairly rudimentary bot that can be easily defeated and level 10 is a very sophisticated bot that uses information not available to players (i.e., a cheating bot). Finally, environmental factors such as fog-of-war were toggled across experiments to investigate their impact.

#### 3.1.3. Game score and reward implementation.

Reward function is an important component of RL and it controls how the agent reacts to environmental changes by giving them positive or negative reward for each situation. We incorporated the reward function for the *TigerClaw* scenario in *StarCraft II* and our implementation overrode the internal game scoring system. The default scoring system in *StarCraft II* rewarded players for the resource value of their units and structures. Our new scoring system focused on gaining and occupying new territory as well as destroying the enemy.

Our reward function awarded +10 points for the BLUFOR crossing the wadi (river) and −10 points for retreating back. In addition, we awarded +10 points for destroying an OPFOR unit and −10 points if a BLUFOR unit was destroyed. In order to implement the reward function, it was necessary to first use the *StarCraft II* map editor to define the various regions and objectives of the map. Regions are areas, defined by the user, which are utilized by internal map triggers that compute the game score, as seen in Figure 4.

Additional reward components can be also integrated based on the Commander’s Intent in the *TigerClaw*, or other scenario, warning orders. Ideally, the reward function will attempt to train the agent to create optimal behavior that is perceived as reasonable by a military subject matter expert (SME).

#### 3.1.4. Deep RL results.

Using the custom *TigerClaw* map, units, and reward function, we trained a multi-input and multi-output deep RL agent adapted from Waytowich et al. The RL agent was trained using the Asynchronous Advantage Actor Critic (A3C) algorithm. In this tactical version of the *StarCraft II* mini-game, as shown in Figure 5, the state-space consists of seven mini-map feature layers of size 64 × 64 and 13 screen feature layer maps of size 64 × 64 × 64 for a total of 20 64 × 64 two-dimensional (2D) images. In addition, it also consists of 13 non-spatial features containing information such as player resources and build queues. The mini-map and screen features were processed by identical two-layer convolutional neural networks (top two rows) in order to extract visual feature representations of the global and local states of the map, respectively. The non-spatial features were processed...
The actions in *StarCraft II* are compound actions in the form of functions that require arguments and specifications about where that action is intended to take place on the screen. For example, an action such as “attack” is represented as a function that would require the $x$ – $y$ attack locations on the screen. The action space consists of the action identifier (i.e., which action to run), and two spatial actions ($x$ and $y$) that are represented as two vectors of length 64 real-valued entries between 0 and 1, which represents the pair of coordinates in the screen where the action should be executed.

The architecture of the A3C agent we use is similar to the Atari-net agent, which is an A3C agent adapted from Atari to operate on the *StarCraft II* state and actions space. We make one slight modification to this agent and add a long short-term memory (LSTM) layer, which adds memory to the model and improves performance. The complete architecture of our A3C agent is shown in Figure 6.

The A3C models were trained with 20 parallel actor-learners using 8000 simulated battles against a built-in *StarCraft II* bot operating on hand crafted rules. Each trained model was tested on 100 rollouts of the agent on the *TigerClaw* scenario. The models were compared against a random baseline with randomized actions as well as a human player playing 10 simulated battles against the...
**Figure 6.** Schematic diagram of the full A3C reinforcement learning agent and its connection to the *StarCraft II* environment representing the *TigerClaw* scenario.

**Figure 7.** Total reward and Blue Force casualties of the trained AI commander (A3C agent) compared to human and random agent baselines. The AI commander is able to achieve a reward that is comparable (and slightly better) than the human baseline while taking a reduced number of Blue Force casualties.

*StarCraft II* bot. Figure 7 shows the plots of total episode reward and number of BLUFOR casualties during the evaluation rollouts. We see that the AI commander has not only achieved comparable performance compared to a
human player, but has also performed slightly better at the
task, while also reducing BLUFOR casualties.

3.2. Case study: RL using OpSim: a military simulator

The TigerClaw scenario, as described in the previous case
study, was also implemented in the OpSim\textsuperscript{54} simulator. OpSim is a decision support tool developed by Cole
Engineering Services Inc. (CESI) that provides planning
support, mission rehearsal, embedded training, and mis-




tion execution monitoring and re-planning. OpSim inte-
grates with SitaWare C4I C2, a critical component of
Command Post Computing Environment (Command Post
Computing Environment (CPCE) webpage: https://peoc3t.army.mil/one\textsuperscript{55,56} and
www.peostri.army.mil/onestaf) (OneSAF)\textsuperscript{55,56} and
MAGTF Tactical Warfare Simulation (MAGTF Tactical
Warfare Simulation (MTWS) webpage: https://coleengi-
tering.com/capabilities/mtws) (MTWS).\textsuperscript{57} OpSim is
designed to run faster than wall clock time, and can run 30
replications of the TigerClaw mission in under 30 s with
modest consumer-grade hardware, which would take
240 h if run serially in real-time. Output of a simulation
plan in OpSim includes an overall ranking of BLUFOR
plans based on criteria such as ammunition expenditure,
casualties, equipment loss, fuel usage, and others. The
OpSim tool, however, was not originally designed for AI
applications and had to be adapted by incorporating inter-
faces to run RL algorithms.

An OpenAI Gym\textsuperscript{58} interface was developed to expose
simulator state and offer simulation control to external
agents with the ability to supply actions for select entities
within the simulation, as well as the amount of time to
simulate before responding back to the interface. The
observation space consists of 17 features vector where the
observation space is partially observable based on each
entities’ equipment sensors. Unlike the StarCraft II envi-
ronment, our OpSim-based prototype of an artificial C2
agent currently does not use image inputs or spatial fea-
tures from the screen images. The action space primarily
consists of simple movements and engagement attacks, as
shown as follows:

- Observation space: damage state, $x$ location, $y$ loca-
tion, equipment loss, weapon range, sensor range, fuel
consumed, ammunition consumed, ammunition
total, equipment category, maximum speed, perceived
opposition entities, goal distance, goal direc-
tion, fire support, taking fire, and engaging targets.
- Action space: no operation, move forward, move
backward, move right, move left, speed up, slow
down, orient to goal, halt, fire weapon, call for fire,
and react to contact.
- Reward function: friendly damaged ($-0.5$),
friendly destroyed ($-1.0$), enemy damaged (0.5),
enemy destroyed (1.0), $-0.01 * \text{km}$ from goal
destination at every step.

3.2.1. Experimental results. Two types of “artificial com-
manders” were developed for the OpSim environment.
The first is based on an expert plan provided as part of
OpSim and developed by Military SMEs using military
doctrinal rules. The second is an RL-based LSTM\textsuperscript{53} deep
neural network with a multi-input and multi-output, which
was trained with A2C algorithm.\textsuperscript{52} OpSim’s custom Gym
interface supports multi-agent training where each force
can use an either rule or learning-based commander.

The policy networks were trained for BLUFOR and
OPFOR on a high-performance computing system with 39
parallel workers that collected 212,000 simulated battles
in 44 h. The trained models were evaluated with 100 roll-
out simulation results using the frozen policy at a check-
point with a rolling average of 195 for the BLUFOR
policy mean reward and $-317$ for the OPFOR policy
mean reward. Analysis of 100 rollout simulations compar-
ing doctrine rule-based and RL-based commanders is pro-
vided in Figure 8. The results show that the RL-based
commanders reduce its own casualties when competing
against a rule-based commander, as depicted in EXPERT
versus RL and RL versus EXPERT plots. Comparing the
RL-based commanders to the doctrine baseline command-
ers, the BLUFOR RL commander, having a stronger
force, succeeds in minimizing BLUFOR casualties from
about 4 to 0.4 and increases OPFOR casualties from 5.4 to
8.4, on average. This outcome is reached by employing a
strategy to engage only combat armor companies and a
fighting infantry company. Here, the BLUFOR RL-based
commander has learned a strategy to utilize its most lethal
units with Abrams and Bradley vehicles while protecting
vulnerable assets from engaging with the OPFOR, as seen in a snapshot of the beginning and end of one rollout shown in Figure 9.

4. Findings and discussion on developing AI systems for C2

Our research highlights several important aspects of developing a training system for AI algorithms in military-relevant games and battlefield simulators for C2, which includes simulation speed, scalability, and adaptability of the simulator, exploitation of the simulator dynamics by the RL agent, diversity of the training data, how human operators would interface with such AI-based systems, and the capability of RL agents to outperform both human and doctrine-based baselines.

First, with respect to both the adapted game and battlefield simulator, we found that an important aspect is simulation speed. RL algorithms are notorious for requiring millions, or even billions, of data samples from the environment to train the best-performing policies, so fast simulation cycles are essential to achieve results in a reasonable time. In our StarCraft II experiments, we were able to simulate 8 million training steps, or 12,800 battles, per day with 35 CPU workers, and that was still not fast enough given that our agents required over 40 million training steps to achieve satisfactory results. Given that the simulations run, in general, for days, another important aspect for a simulator to be considered for this type of research work is scalability. Advances in computing capacity has propelled the field of RL in the past decade, and thus, a simulator for AI research needs to support distributed computing at scale. Yet another aspect is adaptability. The simulator should give the experiment designer the flexibility to model the desired military task, including terrain, assets, and fog-of-war.

Second, with respect to training RL agents in simulators, due to the nature of the exploratory behavior of these
reward-driven algorithms, sometimes learning agents are able to exploit loopholes in simulator dynamics and learn unrealistic behaviors, usually unintended by the simulator designer, in order to maximize the received reward. This type of exploitation is often not detected during training, and is discovered only when carefully investigating the trained policy behavior after the training session. For example, in our early experiments, we found that the agent was able to exploit a simulator deficiency allowing armored assets to cross impassable terrain. In this sense, RL can be a tool to detect and strengthen simulators.

Third, with respect to the training procedure, it should comprise of diverse training data, for instance, terrain, assets, and location, so the learning artificial commander is able to develop a more robust and general policy and be able to address variability that exists in reality. How diverse and how much variability these scenarios should contain is still an open research question. During training, it also helps to evaluate the learning agent periodically, either against a rule or doctrine-based commander or against another learning artificial commander. Once training is complete and the agent is evaluated in test cases, we found that visualization tools are essential to uncover if the artificial commander learned any unintended behavior although more visualization tools are needed to understand each detail of the commander’s actions and plans.

Another challenge is operator interfaces that often fail to gain end-user acceptance. To this end, another capability potentially adapted from the gaming industry is head-mounted displays (HMD) which afford users the ability to interact with content in virtual reality (VR), augmented reality (AR), and mixed reality (MR) settings—collectively XR. While the army has already invested in XR technologies for training, through the Synthetic Training Environment (Synthetic Training Environment (STE) webpage: https://asc.army.mil/web/portfolio-item/synthetic-training-environment-ste/) and to enhance lethality, through the Integrated Visual Augmentation System (IVAS) (Integrated Visual Augmentation System (IVAS) webpage: https://www.peosoldier.army.mil/Program-Offices/Project-Manager-Integrated-Visual-Augmentation-System/), it has relied heavily on advances made in the gaming industry to do so. In fact, many military XR projects are developed using either unity or unreal—the two most popular game development engines. For this project, initial integration has begun to pull live data from the StarCraft II simulation into a networked XR environment. The goal is to allow multiple, decentralized users with the ability to not only examine AI-generated COAs in an immersive platform, but also to interact with the AI and other human decision-makers collaboratively in real-time.

Finally, we found that RL has been able, at least in a set of cases we explored, to outperform both human and doctrine-based baselines without access to prior human knowledge of the task, which becomes even more relevant when task complexity increases and human-dictated rules are more difficult to specify. However, there are still open questions on how human-like the strategies learned are, or how human-like they need to be if that is the case. Another concern is that the reinforcement learned policies are willing to sacrifice assets in the battle if that leads to a higher reward value at the end of the scenario, which may not be an acceptable decision for a human commander. We address this issue when handcrafting the reward function by penalizing the agent for assets lost, however, given that the reward function is also composed of additional goals, this solution is non-trivial to balance in practice.

5. Conclusion

Improving the capabilities of AI algorithms is necessary for mission commanders to keep pace with the increasing velocity and complexity of warfare. We envision the need to develop agile and adaptive AI-support tools for the future battlefield and multi-domain operation scenarios, under the main assumption that the future flow of information and speed of operation will likely exceed the capabilities of the current human staff if the C2 processes remain largely manual. This includes leveraging AI algorithms to analyze battlefield information from multiple sources to correctly identify and exploit emerging windows of superiority.

In this research work, we explore two case studies on modeling a partially realistic mission scenario and training artificial commanders that leverage existing (with some adaptations) off-the-shelf deep RL algorithms. We show that such trained algorithms can outperform both human and doctrine-based baselines without using any pre-coded expert knowledge, and learn effective behaviors entirely from experience. Furthermore, we formulate and empirically confirm a set of important aspects and requirements of developing a training system for AI algorithms in military-relevant games and battlefield simulators in order to serve as a viable platform for research and development of RL tools for C2. Portions of the Related Work, StarCraft II and OpSim case studies also appear in the DEVCOM Army Research Laboratory report ARL-TR-9192.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of
this article: The research was sponsored by the Army Research Laboratory and was accomplished partly under Cooperative Agreement (W911NF-20-2-0114). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the US Government. The US Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation herein.

ORCID iD
Vinicius G Goecks https://orcid.org/0000-0002-4481-671X

References
1. Caffrey MB Jr. On wargaming. Newport, Rhode Island: U.S. Naval War College Press.
2. Perla PP. The art of wargaming: a guide for professionals and hobbyists. Annapolis, Maryland: Naval Institute Press, 1990.
3. Vinyals O, Ewalds T, Bartunov S, et al. StarCraft II: a new challenge for reinforcement learning. arXiv preprint 1708.04782, 2017.
4. Sun P, Sun X, Han L, et al. TStarBots: defeating the cheating level built in AI in StarCraft II in the full game. Corr 2018; abs/180907193.
5. Vinyals O, Babuschkin I, Czarnecki WM, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 2019; 575: 350–354.
6. Waytowich N, Barton SL, Lawhern V, et al. Grounding natural language commands to StarCraft II game states for narration-guided reinforcement learning. In: Artificial intelligence and machine learning for multi-domain operations applications, vol. 11006. International Society for Optics and Photonics, p.110060S.
7. Waytowich N, Barton SL, Stump E, et al. A narration-based reward shaping approach using grounded natural language commands. In: International conference on machine learning (ICML), workshop on imitation, intent and interaction, 2019.
8. Wang X, Song J, Qi P, et al. SCC: an efficient deep reinforcement learning agent mastering the game of StarCraft II. arXiv preprint Arxiv:201213169, 2020.
9. Han L, Xiong J, Sun P, et al. TStarBot-X: an open-sourced and comprehensive study for efficient league training in StarCraft II full game. arXiv preprint Arxiv:201113729, 2020.
10. Bernet C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning. Corr 2019; abs/191206680.
11. Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. arXiv preprint Arxiv:13125602, 2013.
12. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature 2015; 518: 529–533.
13. Hessel M, Modayil J, van Hasselt H, et al. Rainbow: combining improvements in deep reinforcement learning. Proc AAAI Conf Artif Intell 2018; 32: 3215–3222.
14. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. Nature 2016; 529: 484–489.
15. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. Nature 2017; 550: 354–359.
16. Schrittwieser J, Antonoglou I, Hubert T, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 2020; 588: 604–609.
17. Campbell M, Hoane AJ Jr and Hsu Fh. Deep blue. Artif Intell 2002; 134: 57–83.
18. Hsu FH. Behind deep blue: building the computer that defeated the world chess champion. Princeton, NJ: Princeton University Press, 2002.
19. Brown N and Sandholm T. Superhuman AI for heads-up no-limit poker: libratus beats top professionals. Science 2018; 359: 418–424.
20. Gortney WE. Department of defense dictionary of military and associated terms. Technical report, Joint Chiefs of Staff, Washington, DC, 2010.
21. Burns S, Della Volpe D, Babb R, et al. War gamers handbook: a guide for professional war gamers. Technical report, US Naval War College, Newport, RI, 2015.
22. Army U. Army doctrine publication no. 3-0: operations. Washington, DC: Headquarters, Department of the Army, 2019.
23. Kott A. Advanced technology concepts for command and control. Bloomington, IN: Xlibris Corporation, 2004.
24. Kott A, Ground L, Budd R, et al. Toward practical knowledge-based tools for battle planning and scheduling. In: The Eighteenth national conference on artificial intelligence (AAAI-02) was held July 28 – August 1, 2002, at the Shaw conference center in Edmonton, Alberta, Canada pp. 894–899. Menlo Park, CA: American Association for Artificial Intelligence.
25. Ownby M and Kott A. Reading the mind of the enemy: predictive analysis and command effectiveness. Technical report, Solers Inc., Arlington, VA, 2006.
26. Kott A and Corpac PS. COMPOEX technology to assist leaders in planning and executing campaigns in complex operational environments. Technical report, Defense Advanced Research Projects Agency, Arlington, VA, 2007.
27. Kott A, Ludwig J and Lange M. Assessing mission impact of cyberattacks: toward a model-driven paradigm. IEEE Sec Priv 2017; 15: 65–74.
28. Walsh M, Menthe L, Geist E, et al. Exploring the feasibility and utility of machine learning-assisted command and control: volume I, findings and recommendations. Santa Monica, CA: RAND Corporation, 2021.
29. Walsh M, Menthe L, Geist E, et al. Exploring the feasibility and utility of machine learning-assisted command and
control: volume 2, supporting technical analysis. Santa Monica, CA: RAND Corporation, 2021.
30. Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv preprint Arxiv:170706347, 2017.
31. Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation. arXiv preprint Arxiv:150602438, 2015.
32. Boron J and Darken C. Developing combat behavior through reinforcement learning in wargames and simulations. In: 2020 IEEE conference on games (CoG), Osaka, Japan, 24–27 August 2020, pp. 728–731. New York: IEEE.
33. Lanchester FW. Aircraft in warfare: the dawn of the fourth arm. London: Constable Limited, 1916.
34. Washburn A. Lanchester systems. Technical report, Naval Postgraduate School, Monterey, CA, 2000.
35. MacKay NJ. Lanchester combat models. arXiv preprint Math/0606300, 2006.
36. Corps UM. Marine Corps doctrinal publication 1-0. Washington, DC: Marine Corps Operations, 2011.
37. Asher DE, Zaroukian E and Barton SL. Adapting the predator-prey game theoretic environment to Army tactical edge scenarios with computational multiagent systems. arXiv preprint Arxiv:180705806, 2018.
38. Asher D, Garber-Barron M, Rodriguez S, et al. Multi-agent coordination profiles through state space perturbations. In: 2019 international conference on computational science and computational intelligence (CSCI), Las Vegas, NV, 5–7 December 2019, pp. 249–252. New York: IEEE.
39. Zaroukian E, Rodriguez SS, Barton SL, et al. Algorithmically identifying strategies in multi-agent game-theoretic environments. In: Artificial intelligence and machine learning for multi-domain operations applications II, 14–18 April 2019, Baltimore, Maryland, United States, vol. 11006, p. 1100614. Bellingham, WA: SPIE.
40. Asher DE, Zaroukian E, Perelman B, et al. Multi-agent collaboration with ergodic spatial distributions. In: Artificial intelligence and machine learning for multi-domain operations applications II, 27 April–9 May 2020, Online Only, California, United States, vol. 11413. Bellingham, WA: SPIE, p. 114131N.
41. Barton SL, Waytowich NR and Asher DE. Coordination-driven learning in multi-agent problem spaces. arXiv preprint Arxiv:180904918, 2018.
42. Barton SL and Asher D. Reinforcement learning framework for collaborative agents interacting with soldiers in dynamic military contexts. In: Next-generation analyst VI, 15–19 April 2018, Orlando, Florida, United States, vol. 10653, pp. 106503. Bellingham, WA: SPIE.
43. Fu Q, Fan CL, Song Y, et al. Alpha C2 —an intelligent air defense commander independent of human decision-making. IEEE Access 2020; 8: 87504–87516.
44. AlphaDogfight trials go virtual for final event, https://www.darpa.mil/news-events/2020-08-07 (accessed 17 May 2021).
45. AlphaDogfight trials foreshadow future of human-machine symbiosis, https://www.darpa.mil/news-events/2020-08-26 (accessed 17 May 2021).
46. Schwartz PJ, O’Neill DV, Bentz ME, et al. AI-enabled wargaming in the military decision making process. In: Artificial intelligence and machine learning for multi-domain operations applications II, vol. 11413, 27 April–9 May 2020, Online Only, California, United States, p. 114130H. Bellingham, WA: SPIE.
47. Reese PP. Military decisionmaking process: lessons and best practices. Technical report, Center for Army Lessons Learned, Fort Leavenworth, KS, 2015.
48. Spohr L, Soyer H, Munos R, et al. IMPALA: scalable distributed deep-RL with importance weighted actor-learner architectures. In: International conference on machine learning, 10–15 July 2018, Stockholm, Sweden, pp. 1407–1416. PMLR.
49. Kapturowski S, Ostrovski G, Dabney W, et al. Recurrent experience replay in distributed reinforcement learning. In: International conference on learning representations. New Orleans, Louisiana, United States, 6–9 May 2019. ICLR.
50. Liang E, Liaw R, Nishihara R, et al. RLlib: abstractions for distributed reinforcement learning. In: Dy J and Krause A (eds) Proceedings of the 35th international conference on machine learning, proceedings of machine learning research, vol. 80. 10–15 July 2018, Stockholm, Sweden, PMLR, pp. 3053–3062.
51. Narayanan P, Vindiola M, Park S, et al. First-year report of ARL director’s strategic initiative (FY20-23): Artificial Intelligence (AI) for Command and Control (C2) of Multi Domain Operations. Technical Report ARL-TR-9192. Adelphi, MD: Adelphi Laboratory Center, DEVCOM Army Research Laboratory (US), 2021.
52. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: Volume 48: International conference on machine learning, 20–22 June 2016, New York, USA, pp. 1928–1937. PMLR.
53. Hochreiter S and Schmidhuber J. Long short-term memory. Neur Comput 1997; 9: 1735–1780.
54. Surdu JR, Haines GD and Pooch UW. OpSim: a purpose-built distributed simulation for the mission operational environment. Simul Ser 1999; 31: 69–74.
55. Wittman RL Jr and Harrison CT. OneSAF: a product line approach to simulation development. Technical report, MITRE Corp., Orlando, FL, 2001.
56. Parsons D, Surdu J and Jordan B. OneSAF: a next generation simulation modeling the contemporary operating environment. In: Proceedings of Euro-simulation interoperability workshop. 27–29 June 2005, Toulouse, France, Citeseer.
57. Blais CL. Marine air ground task force (MAGTF) tactical warfare simulation (MTWS). In: Proceedings of winter simulation conference, Lake Buena Vista, FL, 11–14 December 1994, pp. 839–844. New York: IEEE.
58. Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym. arXiv preprint Arxiv:160601540, 2016.
59. Amodei D, Olah C, Steinhardt J, et al. Concrete problems in AI safety. arXiv preprint Arxiv:160606565, 2016.
60. Dennison MS and Trout TT. The accelerated user reasoning capabilities development command. Adelphi, MD, 2020.
Author biographies

Vinicius G Goecks is a postdoctoral researcher developing novel machine learning algorithms that learn from human interaction at the DEVCOM Army Research Laboratory, Human Research and Engineering Directorate. He received his Ph.D. and M.S. in aerospace engineering from Texas A&M University, USA, and B.S. in electrical engineering from Universidade Federal de Mato Grosso do Sul, Brazil.

Nicholas Waytowich is a Machine Learning Research Scientist with the DEVCOM Army Research Laboratory currently stationed at Aberdeen Proving Grounds in Aberdeen MD. He received his BS from the University of North Florida in Mechanical Engineering (2010) and his MS in Electrical and Computer Engineering (2013) and his Ph.D. in Biomedical Engineering from Old Dominion University, VA. Using a variety of machine learning techniques, his expertise and research interests include reinforcement learning, human-in-the-loop learning, multi-agent modeling and developing better and more computationally efficient AI by leveraging human interaction.

Derrrik E Asher is a Research Psychologist at DEVCOM Army Research Laboratory in the Computational and Information Sciences Directorate. He is the principal investigator for the Agents Leveraging Learning for Intelligent Engagement with Soldiers (ALLIES) research effort for the Army.

Song Jun Park is a researcher at the Advanced Computing Branch, Computational and Information Sciences Directorate, DEVCOM Army Research Laboratory under the Army Futures Command.

Mark Mittrick is a computer scientist with the U.S. Army Research Laboratory located at Aberdeen Proving Ground since 2003. Prior to that, he received his B.S in Computer Science from Wilkes University in 2003 and later while working at the Proving Ground earned his M.S from Towson University in 2009. He has authored and co-authored several technical reports in the computer science field.

John Richardson is a computer scientist at the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory.

Manuel Vindiola is a Cognitive Scientist at DEVCOM Army Research Laboratory in the Computational and Information Sciences Directorate. He contributes to computational science, reinforcement learning, and neuromorphic computing research for the Army.

Anne Logie is a researcher at the DEVCOM Army Research Laboratory under the Army Futures Command.

Mark Dennison is a cross-reality researcher with U.S. DEVCOM Army Research Lab West supporting the Military Information Sciences Core Competency. He leads an applied research project on providing a cross-reality common operating picture in support of the Army’s Network Cross-Functional Team. He earned a Ph.D. in psychology, an M.S. in cognitive neuroscience and a B.S. in psychology from the University of California, Irvine.

Theron Trout is a researcher at the DEVCOM Army Research Laboratory under the Army Futures Command.

Priya Narayanan is a Research Mechanical Engineer at DEVCOM Army Research Laboratory in the Computational and Information Sciences Directorate. She is the lead for AI for C2 of MDO, an ARL Director’s Strategic Initiative for developing the next generation AI-enabled Command and Control system for the Army.

Alexander Kott is the Chief Scientist of the Army Research Laboratory (ARL), a component of the US Army Futures Command. Earlier he was a Defense Advanced Research Programs Agency (DARPA) Program Manager. He earned his PhD from the University of Pittsburgh, Pittsburgh, PA, in 1989.