From the Game Map to the Battlefield – Using DeepMind's Advanced AlphaStar Techniques to Support Military Decision-Makers

LTC Thomas Doll

LTC Jan-Wilhelm Brendecke
Army Concepts and Capabilities Development Centre
Division I, OR/M&S Section
GERMANY

ThomasManfredDoll@bundeswehr.org
JanWilhelmBrendecke@bundeswehr.org

Matthias Behm

Daniel Kallfass
AIRBUS Defence and Space GmbH
GERMANY

matthias.behm@airbus.com
daniel.kallfass@airbus.com

ABSTRACT

Future warfare scenarios featuring fully digital, AI-assisted command and control and the use of unmanned systems will have a dramatic impact on the tempo of combat operations. Consequently, they will put the cycles of military decision-making under even higher time pressure. Modelling and simulation in combination with advanced AI techniques will become key enablers for future decision-support systems. These systems will support military decision-makers in the assessment of threats as well as in developing and evaluating the best possible courses of action for their own forces. The latest developments by AI research companies in the civilian domain, such as DeepMind's AlphaStar, have applied advanced deep reinforcement learning techniques to popular games like StarCraft II to train RL agents to develop superior strategies for beating their opponents.

This paper presents the results of a study conducted by the Army Concepts and Capabilities Development Centre and Airbus. The aim of the study was to evaluate how the aforementioned machine-learning techniques can be adapted and employed to train an RL agent capable of acting as a battalion commander in a combat simulation ("ReLeGSim"). In each time step of this simulation, the RL agent can send orders to the available units/companies or request multi-domain fire support. The "ReLeGSim" simulates the behaviour and combat attrition of each company/unit and fire support element at the level of the individual platform. It then returns feedback (so-called reward) to the RL agent in order to assess and improve its behaviour during the training cycle. It is also possible to select multiple trained RL agents and let them play against each other in a league system to further improve them.

Having undergone such training, the RL agent can be applied to an actual scenario. The resulting strategies can be proposed to the battalion commander as possible courses of action in a decision cycle.

1.0 INTRODUCTION

The basic ideas, processes, and technologies for training an Artificial Intelligence (AI) are freely available. Much of the development in this area takes place publicly and is made available via the Internet. In particular, 'human versus computer' games are often used to demonstrate progress in AI development. The used games reach from classic board games such as Chess and Go to modern computer games such as Quake and Dota 2. For this study, the achievements around the computer game StarCraft II are of particular interest.

StarCraft II is a real-time strategy game in which the players have to produce and control units with different capabilities. A player only sees what is within his own units' visibility range. The final objective of the game is to defeat the opponent player. Therefore, the optimum employment of the units has to be coordinated

while resources are limited and information is incomplete. From an abstract perspective, some parallels become apparent between the game StarCraft and a military battlefield simulation. DeepMind, a subsidiary of Alphabet (Google), managed to train a Reinforcement Learning (RL) agent using an algorithm called AlphaStar, which masters the game at the "grandmaster level". Human players do no longer stand a realistic chance against the RL agent and are beaten in 98.8 % of the games.

Unlike traditional rule-based approaches, RL has some advantages when dealing with military situations. The information space in a military environment is usually scattered and fragmented. The number of possible combinations of friendly and enemy units, terrain characteristics and infrastructure is vast. A lack of all-embracing Joint Intelligence, Surveillance and Reconnaissance (JISR) as well as technical failures, sabotage, cyber-attacks and hostile camouflage, deception and jamming measures result in an incomplete picture of the battlespace and a high amount of uncertainty. Facing such an information space, rule-based approaches reach their limits. Thus, RL-based approaches might enable developers and system architects to implement more reliable AI systems for use in a military environment.

The described study, conducted by the German Army Concepts and Capabilities Development Centre (CCDC), aims at adapting AlphaStar and other successful RL training techniques for use in a battlefield simulation. From a technological view, this means qualifying an RL agent to develop courses of action for a broad spectrum of situations and to evaluate their respective chances of success. A constructive agent-based battle simulation is used as an underlying simulation framework. The trained AI agent represents the command level of a battalion. The study intends to answer the question whether the RL approaches used in the game industry can be utilized to support a military troop leader during the decision-making process.

2.0 RELATED WORK

In recent years, Artificial Intelligence has been used to solve complex tasks or tasks that were impossible to solve by manually designed algorithms. Lately, Reinforcement Learning has become increasingly popular by yielding outstanding records and superhuman performance. In particular, the game industry utilized sophisticated RL techniques for games like Dota2 or StarCraft. The aim of this study is to adapt those RL techniques to a sophisticated combat simulation in order to create superhuman decision-support for military operations.

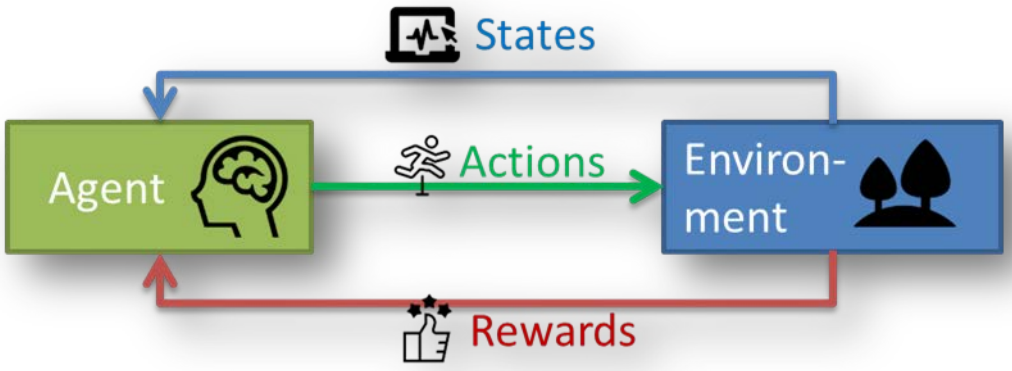

Figure 2-1: RL schematic representation

In RL, the agent is an AI entity that learns a policy, a sophisticated pattern of behavior based on an observed environment. The policy – the agent's strategy – is represented by a policy network that is trained with a policy algorithm. The policy network outputs are the possible actions of the AI agent (Actions). Performed

actions are sent to the environment and, as feedback, the agent receives rewards and observes a new environment state (Rewards, State), the so-called observation space (see Figure 2-1). The reward is a function that needs to be designed by the developer. It depends on the desired agent's objective and the environment in which the agent is embedded in. In a military context, the state might include own and enemies unit positions, weather effects or target coordinates.

At every time step, the agent conducts an action, resulting in a possible change of the observation space. Based on the change, the agent estimates a value function. It represents the cumulative expected discounted reward the agent will receive by following the current policy at the given state. In general, the higher the value the better. It reflects the importance of the current state to reach the overall goal. A complete simulation of a scenario is called an episode. At the end of each episode, the reward and the value function are used to compute a loss. The loss reflects how good the current policy is. By minimizing this loss, the agents will optimize the policy resulting in an optimized behaviour. During training, those cycles are performed repeatedly until a specific number of episodes is run or the desired loss is reached.

In 2013, Mnih et al. reached human-level intelligence with an AI implemented to play Atari video games [1]. In 2015 and 2016 the used training algorithms were improved, resulting in an asynchronous method called ‘vanilla policy gradient’ [2][3]. In 2016, DeepMind revealed AlphaGo, an AI capable of playing Go at a professional level [4]. In October 2017, an improved version of AlphaGo that did not use any human data was published, followed by a more generalised algorithm called AlphaZero in December 2017 [5][6]. AlphaZero was a remarkable advance in RL research since it can master three games, Chess, Shogi and Go, only by being provided with the game rules. AlphaZero developed unconventional strategies that enabled it to beat the world champions of each of the three games. In 2019, DeepMind (AlphaStar) and OpenAI (Dota 2) demonstrated that AI can also outperform human players in online video games [7][8].

3.0 RELEGS – REINFORCEMENT LEARNING FOR COMPLEX COMBAT SITUATIONS

3.1 The Simulation Environment “ReLeGSim”

For StarCraft II, the game manufacturer Blizzard has published a wide range of information and data, aiming on advancing the research and development of AI. For example, nearly half a million anonymised replays of played games were made available. In addition, Machine Learning APIs were made available for publicity. For AlphaStar, not only was the simulation environment already in place and complete. Moreover, there were also framework-like setups that accelerated the RL process enormously. In contrast to AlphaStar, a simulation environment did not exist at the start of the ReLeGs study. It was therefore a huge challenge to create a simulation environment that enables successful AI training in a military setting. The simulation environment plays a key role in training and testing RL agents. Multiple requirements had to be met to ensure successful training.

The developed simulation environment, ReLeGSim, represents two opposing forces at a battalion level. The two sides represent attacker and defender in the scenario. The attacker has the overall objective of taking a defined area while the defender has the task of holding it. Both sides can give commands to company-level elements. The companies in turn contain platforms consisting of subcomponents such as sensors and weapon systems. Different platform types such as reconnaissance units, battle tanks, or infantry fighting vehicles with different capabilities depending on the initial situation on the battlefield can be deployed. In addition, both sides can request joint fire support elements such as combat helicopters, artillery, or close air support. Furthermore, they have the option to lay minefields. Like AlphaStar, there is no God View, but only partial knowledge obtained through the sensors of the players' units. Thus, ‘Fog of War’ is implemented. Both sides must actively explore the environment to gather and update their own situational awareness.

For successful RL training, the agent has to experience as many different situations as possible to adapt and develop successful strategies. For this, ReLeGSim implements a procedural level generator to generate randomised scenarios. The terrain is generated either randomised or based on real-world data (Open Street Map - OSM). All scenarios contain elevation data that can affect visibility and weapon effects. The elevation data is derived from real-world elevation models (e.g., SRTM) or generated through algorithmic solutions. The number of own and enemy companies and platforms can also be randomly generated or specifically defined. Additionally, during training or testing of agents, real-world situations can be integrated. The initial situation of each scenario in ReLeGSim consists of terrain, elevation model, and an initial location of the units.

For cluster operation and training, ReLeGSim is parallelizable and headless executable. This means that the simulation core can run independently from the Graphical User Interface (GUI). The batch operation and the parallel execution in combination with the possibility to run ReLeGSim significantly faster than real-time (speed ReLeGSim > 600 * real-time) ensures that the simulation does not represent a bottleneck for the RL training. ReLeGSim is based on the scripting language Python and is therefore accessible and usable for almost all RL training environments that meet the Gym standard [9].

Once the scenario is generated, the ReLeGSim can be started. Inputs by humans or RL agents are made in discrete time steps, such as in turn-based games. The inputs are then executed continuously by the underlying simulation model up to the next time step. For instance, an RL agent specifies a position for a company. When executing the step, the selected company will continuously move towards the desired position until the simulation reaches the next discrete time step. These temporal boundary conditions allow the RL agents to act in discrete time steps. The length of the time steps has a direct influence on the needed training resources as well as on the training performance. A higher clock rate requires significantly more training resources and thus usually increases the training duration.

The selection of target positions and target units as well as the deployment of combat support elements in ReLeGSim is done on a two-dimensional grid. For this purpose, the entire terrain and elevation model was rasterised with a previously defined resolution. For example, a human player can command a company to move to coordinate 10/10. The goal of this implementation was to reduce the complexity of the model. Figure 3-1 depicts the rasterization of the selected terrain as well as the associated elevation model.

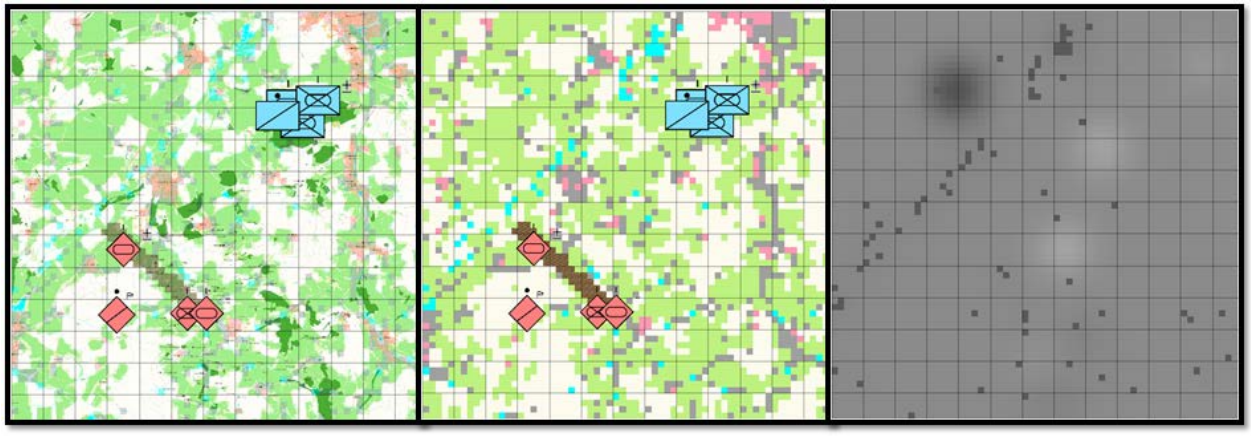

Figure 3-1: ReLeGSim rasterisation process. Open Street Map (left), rasterised terrain (middle), elevation model (right)

ReLeGSim can be run in single-player, multiplayer, or replay mode. With these different configurations, both single-agent and multi-agent training approaches can be realised. The replay mode is especially useful

for evaluating the quality of the trained models. Saved replays can be used for supervised learning. Initially, only the attacker (BLUE) was trained using RL. Since there was not enough replay data available to learn the defender's behaviour, it was implemented in form of rule-based algorithms. Later, this solution was replaced by a trained red RL agent.

3.2 ReLeGs AI Architecture Overview

The architecture of the ReLeGs AI model is based on the solution used by Deepmind's AlphaStar. First, scalar and visual inputs are encoded separately. The results of those encodings are concatenated and made available to a memory component. After this, the actions for the current step are computed. The value network receives the concatenated output of the memory component with current game statistics regarding the enemy situation. Figure 3-2 shows the architecture of ReLeGs. However, for ReLeGs significantly less computing power was available. Thus, the architecture had to be reduced and simplified.

The observation space in ReLeGs consists of scalar and visual information. The scalar input represents the metadata of the current environment state. This includes, for example, company and platform parameters, line-of-sight and effective ranges, and terrain properties such as elevation data. In addition, partial knowledge of adversarial units is encoded. The scalar values are normalised and passed to an entity encoder. Initially, this was done with a Multi-Layer Perceptron (MLP). Later in the study, this element was replaced by a Multi-Head Attention (MHA) network, which increased the quality of the trained agents [10].

In the visual input of the observation space, the rasterised terrain and elevation model as well as the units are encoded. Different properties of the terrain such as forest, mountain, or river are encoded using a previously defined colour scheme. This part of the architecture, the so-called spatial encoder, consists of a Convolutional Neural Network (CNN). Initially, already existing CNN networks such as MobileNetV3 and ResNet-50 were used. The results of the training with these partly pre-trained networks were not satisfactory. Therefore, a custom CNN was implemented during the study. It is significantly smaller and less computationally expensive compared to the one mentioned previously. The reduction of the model's size and complexity resulted in improved training performance.

After encoding the visual and vector inputs, the outputs of the entity and spatial encoders are concatenated and passed to the core module. This core consists of a Long Short-Term Memory (LSTM) component. This element is used to address the long-term planning problem [11]. Among other things, the agent has the task of requesting combat support based on the current situation available to him. Depending on the type, these combat support elements require up to 15 minutes until they have an effect in the scenario. The forward-looking strategic planning of these high-value assets is realised with the aid of the LSTM. A skip connection was also integrated for this purpose. This connection serves to better train the upper layer. The output of the LSTM and the elements of the skip connection are concatenated and made available as input for the action heads.

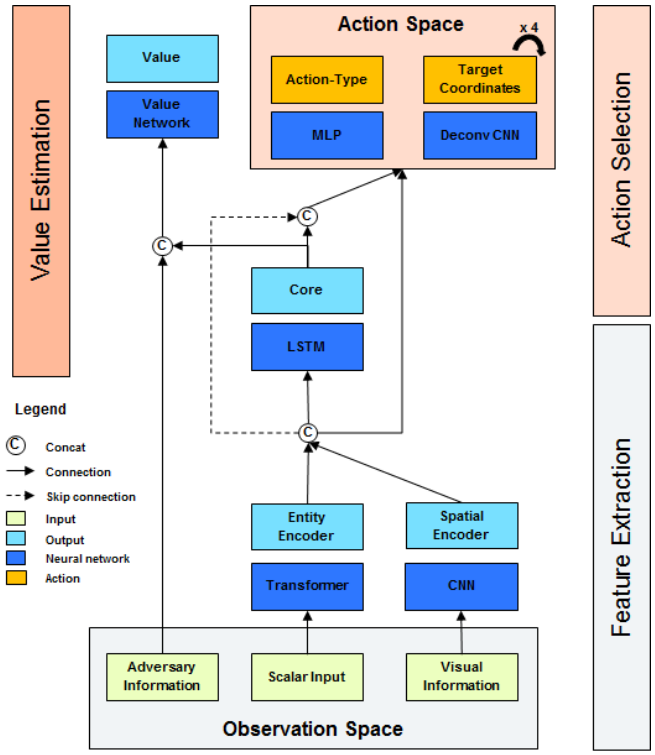

Figure 3-2: ReLeGs AI Architecture

The action heads consist of different MLPs, one per type of entity. The output of each of these MLPs is one of the following possible actions: 'do nothing', 'cancel current action', or 'request new action'. Invalid actions are masked before the action is sampled. The masking of invalid or impossible actions can lead to significantly better training results. If the selected action requires a position, coordinates are sampled afterward using the location head. This was implemented as an upscaling network, which is shared by all units irrespective of their type. At this point, masking of invalid coordinates also takes place. The creation of target coordinates is then done independently for all unit types according to their specific heads. There is one head each for regular units, reconnaissance units, combat support elements, and for the use of mines. The latter is only available for the defending player, as the attacker cannot use mines in the current implementation. If there are several units of the same type in the scenario, they all use the same head.

For the value network, the output of the LSTM is concatenated with information on the adversary and specific game statistics. Subsequently, the outcome of this operation is passed to an MLP. The result is an estimate of the value function. The value function represents the cumulative expected discounted reward the agent will receive by following the current policy at the given state. Generally speaking, the higher the value the better. It reflects the importance of the current state to achieve the overall goal. The accurate approximation of the value function is therefore essential for training success.

3.3 Action Space

As mentioned above, there are three different types of actions: 'do nothing', 'cancel current action', or 'request new action'. In case the agent decides to generate a new action, the AI will calculate coordinates for potential new positions for each unit or for the deployment of combat support elements. If 'cancel action' is selected, the network responsible for generating and selecting coordinates is not called. Instead, 'do nothing' is sent to and executed in the simulation.

As with AlphaStar, the action space in ReLeGs is large. The RL agent in ReLeGs must choose from about

58.000 different actions at each discrete time step. Such a large action space can lead to very large model sizes. To keep the size of the model manageable, some parts of the model were shared between entities of the same type. In addition, an upscaling network was used to generate the coordinates. This saves a huge amount of parameters and helps to keep the model small.

For the generation and selection of the actions, an autoregressive action scheme is implemented. This method was used to generate actions that are dependent on other actions. For example, the selection of coordinates considers the previously selected action. This approach has led to several difficulties, which resulted in a restructuring of the model. As a result, the interdependencies between the actions were removed. The calculation of the action and coordinates was then implemented independently of each other and in parallel. For this part, different heads were implemented for the action and coordinates.

3.4 Reward

Designing the reward function is an elementary and difficult part of RL. The performance of the model and the goodness of the reward function are closely related. During training, the agent is going to optimize his results by optimizing the estimated reward. The more specific the reward function is, the more it will influence the training of the model. This can lead to a poor and undesired behaviour of the agent.

To illustrate this idea, let us take a closer look at the reconnaissance unit. It has a greater sensor range than other units but very limited combat capabilities. These kinds of units are usually used to reconnoitre the battlefield from a distance, and usually do not actively engage in combat. If this unit was not explicitly considered in the reward function, the agent could develop the behaviour not to use the reconnaissance unit at all (since its combat performance is rather poor) or just to send it directly towards the opponent without any support.

The reward has to be designed to reach the desired strategy and avoid unwanted behaviour. As explained in the example above, the reward function can heavily influence the training of the agent. Finding the perfect balance is not easy. To create a generic behaviour not influenced by predefined rules, the reward should be designed as abstract as possible. In this study the best results were obtained with a combination of rewards: win/loss of combat, reaching the target area, successful engagement (hit) of target units with combat support and additional win/lose at the end of the episode.

4.0 EXPERIMENTS & RESULTS

4.1 Experiment Setup

Deepmind´s AlphaStar was able to defeat the built-in “Elite” level AI of StarCraft II without any RL by only applying supervised learning techniques. They introduced RL techniques to improve the agent’s performance even further, which eventually led to convincing victories against professional human players. The idea was to first train the model to learn human strategies and then extend them with RL to achieve superhuman performance. To do so, they used a set of recorded games from StarCraft II players. The goal was to reproduce games played by humans to teach the model human behaviour. This represented another challenge for this study. With the help of the replay mode in ReLeGSim it is possible to record episodes played by human players for later use in supervised learning. However, due to the time constraints of the study, this would have only been possible for a relatively small number of episodes (<100) and therefore not sufficient for a supervised learning phase.

ReLeGs uses the library RLlib, currently the most popular open-source library for Reinforcement Learning. It contains a huge set of implementations of state-of-the-art RL techniques [12]. The API provided by RLlib

was used to set up the training environment for the model. Through RLlib the custom simulation environment, reward function, and the model could be easily integrated.

In a next step, the training algorithm was selected. RLlib offers a wide range of algorithms. The best results were obtained with Asynchronous Proximal Policy Optimization (APPO), but also Importance Weighted Actor-Learner Architecture (IMPALA) provided good results. These two algorithms were fast and efficient in training considering the hardware limitations. Due to the classification of the data used in ReLeGSim, no public cloud such as Google Cloud Platform (GCP) or Amazon Web Services (AWS) was used.

As hardware, the latest Nvidia RTX 3090 GPUs were used. These were integrated into a cluster network and formed a private cluster on which the RL agents were trained. Also, APPO and IMPALA algorithms were used because of the disproportionate CPU power available compared to the GPU power available. The goal was to sample asynchronously and to parallelize the rollout workers. Rollout is a key concept in RL. It consists of simulating multiple complete or truncated episodes in parallel with the current policy at a certain time step. These rollouts are put together and optimised at the same time. The rollout technique helps to simulate and to evaluate as many as possible next available actions in order to choose the best one in the end. Through this approach, the best strategy can be learned by considering uncertainty. Here the uncertainty comes from the opponent's behaviour. Thus, trying every possible action multiple times helps to evaluate the opponent's reaction to those actions and in the end, helps to choose the best action to take in a given situation.

Each rollout worker ran multiple instances of the same agent in a ReLeGSim environment. The combination of asynchronous sampling and parallel running rollout workers allowed a high sample throughput of several thousand samples per second. Figure 4-1 depicts the training architecture of ReLeGs.

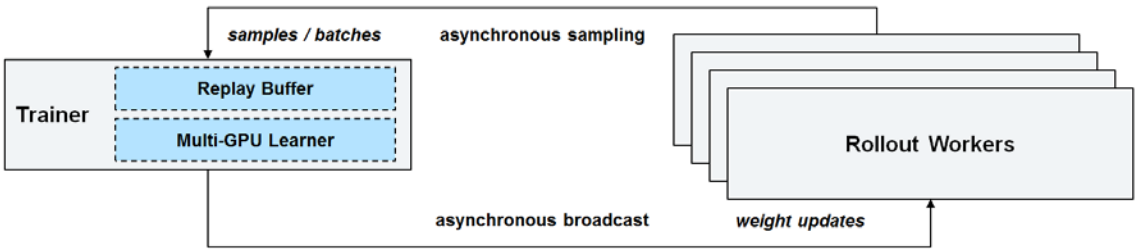

Figure 4-1: ReLeGs training architecture adapted from RLlib [12]

Several other techniques were used during training, too. They determine the efficiency of the training. First, a simple Grid Search was used to find a suitable parameter configuration before moving to Population Based Training (PBT). PBT is a method that finds the best set of parameters by exploiting a population while also randomly exploring new sets of parameters. This alternation between exploitation and exploration helps to quickly find the best set of hyperparameters.

4.2 Pre-Training

As mentioned in the previous chapter, there were not enough human replays available to apply supervised learning successfully. To cope with that problem, it was tried to artificially generate replays. This approach aimed at teaching the RL agent human strategies to serve as a basis for RL training. For this purpose, algorithmic solutions were developed for the RED and BLUE players. These were applied in randomised scenarios and recorded as replays. The replays served as a baseline for a supervised learning phase in which behaviour cloning was used [13]. Unfortunately, the results of this pre-training were not satisfactory. The

agent focused too much on the algorithmic solutions and hardly learned any new strategies during the subsequent RL. Due to this, behaviour cloning was not used in the later stage of the study.

In the further course of the study, parts of the network underwent pre-training. The CNN was trained using an auto-encoder/decoder. Several hundred thousand training data sets were generated and used during the process. This approach made it possible to design the CNN to be as small as possible while still being able to recognize and extract all the required features. The same approach was used to train the RL agent to understand the Fog of War. Solving the 'Fog of War' problem is usually very computationally intensive since it uses a calculation in a three-dimensional space. It was tried to train the RL agent to calculate the Fog of War itself by including the information of the elevation model and the sensor range. This approach is depicted in Figure 4-2. Represented on the left side is the actual CNN input, which comes from the observation space. The middle image displays the algorithmically generated training data set. Depicted on the right side is the output of the CNN.

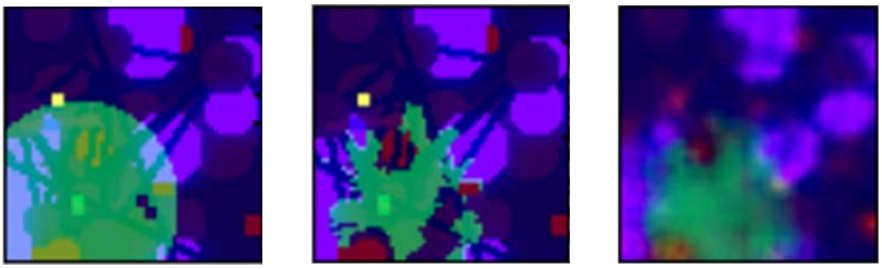

Figure 4-2: CNN auto-encoder/decoder Fog of War

A major benefit of this approach was the verification of the CNN parameters. The network architecture was optimised and the spatial resolution improved. A direct comparison of the training duration between a completely untrained model and a pre-trained CNN is not completely possible. However, a performance boost was observed in the less complex levels at the beginning of the training. No significant difference could be detected over the entire training period.

Additionally, weight initialisation using an Auto-Encoder for the MLP was implemented. The results, however, were not satisfactory: the data from the simulation were changing too fast and the initialised weights had little to no effect.

4.3 ReLeGs Empirical Evaluation

To evaluate the quality of the trained RL agents, test scenarios were created. These scenarios are based on tactical situations that are close to reality and are used in the tactical training of military officers. Specific quality metrics were created to indicate how well the learned behaviour is performing. These metrics include duration of engagement, casualties on both sides, use of reconnaissance assets, and mission success.

To win a scenario, different mission objectives must be achieved depending on the mission. The attacker has the goal to take a defined area and destroy all enemy units in it. The defender has the goal to hold the area and repel any attacks. For the empirical evaluation, the scenarios were replicated several hundred times to negate stochastic effects and ensure the statistical significance of the results. The mean win rate, which could be derived from the results, is an empirical metric that allows drawing direct conclusions about the quality of the model. Besides the reward, this value is the most important indicator for the quality of trained RL agents in the study.

In addition to the test scenarios, which were only created for the evaluation of trained RL agents, an empirical evaluation also took place during the training. With the help of self-defined quality metrics, which were tracked during the training, a rough quality value could be determined for each model. Since the simulation environment progressively increases the complexity of the generated scenarios during training, quality metrics could be tracked straightforward with the help of custom loggings in TensorBoard.

With help of the procedural level generator, it was possible to ensure that the RL agent could train in scenarios that increase in complexity over a training period. The increasing complexity included the terrain layout, such as the number and size of mountains, water and urban terrain or the number of simulated entities represented. So far, it has been possible to successfully train with up to ten companies and four different kinds of joint fire support elements for each side on many different levels with increasing complexity.

4.4 Single Agent Training

The initial focus of the study was on single-agent training with the attacking player being trained. The opponent (defender) acted based on a rule-based behaviour. With the help of PBT and the resulting hyperparameter configuration, the trained RL agent reached the most complex scenario level provided by the procedural level generator in a short amount of time. The trained agent seems to have developed an understanding of different unit types. Specifically, it uses reconnaissance units to master the Fog of War. In addition, it uses the available joint fire support elements to destroy the reconnoitred enemy.

For training, APPO and IMPALA were used (see Ch. 4.1). APPO provided the best results. During the PBT at least 10 samples ran in parallel. After a certain training period, the hyperparameters of the best sample were adopted for further training. As a metric to determine the quality of the sample of the PBT training, a combination of the mean win rate across all levels and the achieved reward was used.

4.5 AI vs. Human

As a part of the study, the RL agent's learned behaviour was tested against a human counterpart. A military officer experienced in military tactics operated the human side and competed against ReLeG's AI. The first experiences showed that the RL agent uses unexpected behaviour. Sometimes it moved heavy tanks through wooded areas and circled the defender to attack him from behind or into the flank. In some cases, this behaviour was successful. It is not clear yet if that behaviour is due to superhuman cognition of the RL agent or due to the modelled scenario (especially terrain). However, even in the latter case, the AI would be superior to the human-based on the simulation. The question arises whether this would also be the case in a more detailed simulation or within the real world.

From a human perspective, the RL agent rarely chooses the most likely course of action for his attack. This may be due to the RL agent trying to beat classic tactics and, for example, avoid minefields. Further, the AI agent was not limited in its movement to the left or right since there were neither neighbours nor defined areas of operation limiting the freedom of manoeuvre.

The result of the AI vs. human test also showed that the rule-based behaviours of the defending side do not provide enough variation and complexity to challenge the attacking RL agent. Thus, the use of basic human tactics usually resulted in success and defeated the RL agent. The quality of the behaviour and strategies of the trained agent is largely defined by the quality of the opponents available to him during training. Therefore, in the further course of the study, the defending side will also be trained to ensure an appropriate variability of behaviour on both sides.

4.6 League – AI vs. AI

For AlphaStar, DeepMind extended the PBT approach with a league system. Using a variety of different

approaches, a whole league of RL agents with different strengths and weaknesses was generated. This training method is useful to pinpoint the agent`s weaknesses and consequently train the agent to overcome them. Lei Han et al [14] has further optimised the DeepMind approach to require significantly fewer resources and computing power.

This successful league approach of DeepMind and the modifications of Lei Han et al. will be adapted in the further course of the study. It is planned to populate the league with different agents, resulting from the PBT of both sides. The expected outcome is to train an RL agent with a very diverse league setup and enable it to master as many different strategies as possible and use them to beat its opponent in any given scenario.

5.0 CONCLUSION

The study did not aim at creating an all-inclusive artificial intelligence to take over decision-making as a whole and make human intervention obsolete. On the one hand, this cannot be achieved with the technology available currently, and on the other hand, such an approach would contradict the ethical and moral concept of military command responsibility in the German Armed Forces. Human control must always be ensured. “Man makes decisions while the machine provides support!”

During the study, it quickly became clear that DeepMind's AlphaStar approach cannot simply be copied. On the one hand, it turned out that the methods used were not all completely disclosed after all, on the other hand, the resources available in the study do not even come close to the computing capacity that DeepMind had available for AlphaStar. In order to be able to carry out the intended training in a target-oriented manner, both the simulation environment and the reinforcement-learning framework, i.e., the RL agent itself had to be simplified and optimized significantly in terms of computational speed. This made it possible to reduce the complexity of the system and ultimately limit the computing power required. Nevertheless, the training of the RL agent turned out to be highly complex, which ultimately led to the fact that both, the training environment and the structure of the RL agent, had to be continuously adapted and optimized.

While the training phase of the ongoing study has not been completed yet, the RL agent has already produced quite remarkable behavioural patterns. For example, it employs tank companies mainly in open terrain. It uses long-range weapon systems at an early stage to influence the balance of power in its favour. In addition, it tasks remote JISR units to ensure early detection and engagement of enemy forces.

For movements of friendly forces, the RL agent exploits terrain outside the effective ranges of enemy weapon systems. Since no such information had been given, the agent must have 'learned' the ranges of the enemy’s individual weapon systems. In another scenario, the RL agent purposefully slowed down follow-on forces by employing minefields, thus preventing them from closing up to advance elements.

It must be noted in this context that these behavioural patterns can be perceived by humans while they watch and evaluate the RL agent’s activities. The RL agent is not familiar with principles such as delay, formation of reserves, counterattack and local hasty counterattack. At any given time, the RL agent calculates probabilities for the available courses of action and acts accordingly. Due to the complexity of the setting, it is not (yet) easy to really comprehend in a human-understandable way how these probabilities come about. The explanation of the RL agents’ behaviour can be further researched through the very field of study “Explainable AI“(XAI).

In the course of the still ongoing study, further behavioural patterns are expected to develop, such as the prioritisation of targets when using long-range weapon systems or the formation of reserves to quickly provide reinforcements at the point of main effort or even to shift it as necessary.

Findings so far already suggest that the approach presented has great potential of enabling an RL agent to

perform complex tactical tasks in an automated manner. Supporting decision-making is just one potential application. Others include the training of automated unmanned systems or the use of computer-generated forces in simulation-based training.

6.0 WAY AHEAD

Many national and international studies indicate that future operations at all levels will be conducted at a much higher pace. While in the past a brigade commander had several hours to make a decision, this timeframe will have to be increasingly and significantly reduced to achieve superiority. There are many different reasons for this. One of them is obviously the use of unmanned aircraft systems (UAS) outfitted with sensor payloads and organic weapons as could be seen in the recent conflict in Nagorno-Karabakh.

Another driver of this development is the increasing digitisation and networking of all platforms, sensors and effectors on the battlefield enabling a significant reduction of the response times between detection and engagement. Given the short distance between effector and target, the time of flight of the delivery means is almost negligible when loitering ammunition is used. Thus, firing sequences that took several minutes with traditional artillery systems will be possible within seconds.

The use of hypersonic weapons or improved satellite-based target acquisition capabilities, too, are precursors of accelerated combat operations. The same applies to the employment of long-range weapon systems bringing pinpoint effects to bear, not to mention the use of swarms of tactical drones as future weapons against all kinds of armoured combat vehicles. This list of reasons is by no means exhaustive.

All of this suggests that the effectiveness and superiority of modern armed forces will hinge on an accelerated decision-making process. The results of the study at hand establish that Reinforcement Learning clearly has the potential to bring about the required time savings. It also reveals, however, that considerable effort will be required to fully exploit this potential.

This investigation was focused on developing the reinforcement learning framework and the training itself. Even though expedient tactical behaviour became apparent, the training environment calls for a much higher level of detail to allow for this technology to be operationalised. This will require adequate hardware and continually optimised software, i.e., the simulation and training environment to train the RL agent and optimise its architecture. To that end, it is necessary to have developers with technical experience in the relevant AI processes as well as well-seasoned military tacticians with the necessary basic technical understanding for applied AI methods. National expertise to build future military AI capabilities will continue to grow in importance.

7.0 ACKNOWLEDGEMENT

We would like to thank the entire ReLeGs study group and especially LTC Dr. Dietmar Kunde from the German Army Headquarters and MAJ Stefan Göricke from the Army Concepts and Capabilities Development Center for their very valuable contributions to the study. We would also like to thank Michael Möbius and Sebastian Steinmann from Airbus for their invaluable support during the ReLeGSim and RL implementation.

8.0 REFERENCES

[1] Mnih, Volodymyr & Kavukcuoglu, Koray & Silver, David & Rusu, Andrei & Veness, Joel & Bellemare, Marc & Graves, Alex & Riedmiller, Martin & Fidjeland, Andreas & Ostrovski, Georg & Petersen, Stig & Beattie, Charles & Sadik, Amir & Antonoglou, Ioannis & King, Helen & Kumaran, Dharshan & Wierstra, Daan & Legg, Shane & Hassabis, Demis. (2015). Human-level control through deep reinforcement learning. Nature. 518. 529-33. 10.1038/nature14236.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236

[3] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. arXiv:1602.01783 [cs]. http://arxiv.org/abs/1602.01783

[4] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484–489. https://doi.org/10.1038/nature16961

[5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. Nature, 550(7676), 354–359. https://doi.org/10.1038/nature24270

[6] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv:1712.01815 [cs]. http://arxiv.org/abs/1712.01815

[7] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z

[8] OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d O., Raiman, J., Salimans, T., … Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680 [cs, stat]. http://arxiv.org/abs/1912.06680

[9] OpenAI. (o. J.). Gym: A toolkit for developing and comparing reinforcement learning algorithms. Retrieved on 24th of August, 2021 from https://gym.openai.com

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv:1706.03762 [cs]. http://arxiv.org/abs/1706.03762

[11] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[12] RLlib: Scalable Reinforcement Learning—Ray v2.0.0.dev0. (o. J.). Retrieved on 24th of August, 2021, from https://docs.ray.io/en/master/rllib.html#rllib-index

[13] Goecks, V. G., Gremillion, G. M., Lawhern, V. J., Valasek, J., & Waytowich, N. R. (2020). Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. arXiv:1910.04281 [cs, stat]. http://arxiv.org/abs/1910.04281

[14] Han, L., Xiong, J., Sun, P., Sun, X., Fang, M., Guo, Q., Chen, Q., Shi, T., Yu, H., Wu, X., & Zhang, Z. (2021). Tstarbot-x: An open-sourced and comprehensive study for efficient league training in starcraft ii full game. arXiv:2011.13729 [cs]. http://arxiv.org/abs/2011.13729