Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

AI-BASED MILITARY DECISION SUPPORT USING NATURAL LANGUAGE

Michael Möbius
Daniel Kallfass

Airbus Defence and Space GmbH
Dept. Operational Analysis and Studies
Claude-Dornier-Strasse
88090 Immenstaad, GERMANY

Thomas Doll

Army Concepts and Capabilities Development
Centre, Division I, OR/M&S Section
Bruehler Strasse 300
50968 Cologne, GERMANY

Dr. Dietmar Kunde

German Army Headquarters
von-Hardenberg-Kaserne
Prötzeler Chaussee 25
15344 Strausberg, GERMANY

ABSTRACT

To mimic a realistic representation of military operations, serious combat simulations require sound tactical behavior from modeled entities. Therefore, one must define combat tactics, doctrines, rules of engagement, and concepts of operation. Reinforcement learning has been proven to generate a broad range of tactical actions within the behavioral boundaries of the involved entities. In a multi-agent ground combat scenario, this paper demonstrates how our artificial intelligence (AI) application develops strategies and provides orders to subsidiary units while conducting missions accordingly. We propose a combined approach where human knowledge and responsibility collaborate with an AI system. To communicate on a common level, the orders and actions imposed by AI are given in natural language. This empowers the human operator to act in a human-on-the-loop role in order to validate and evaluate the reasoning of AI. This paper showcases the successful integration of natural language into the reinforcement learning process.

1 INTRODUCTION

Military decision support is a major research field for future combat that can be supported by a wide range of solutions. One instance could involve using combat simulations for course of action analysis to evaluate one's own options or the potential options of the opponent prior to a planned campaign. To create a serious combat simulation, the representation of military operations must be realistic to a considerable extent. This includes correctly modeling the physical properties and capabilities of tactical entities, as well as the tactical behavior, rules of engagement, and concepts of operation.

The scripting of an operation leads to a representative stochastic outcome in which one's own forces act as planned. However, scripting the opposing forces seldom fully meets the expected behavior. Therefore, disruptive outcomes rarely occur.

The latest publications on advanced artificial intelligence (AI) applications such as DeepMind's AlphaStar or Defense Advanced Research Projects Agency's (DARPA) Alpha Dogfight have shown that deep reinforcement learning (deep RL) is capable of developing superhuman strategies by learning and

optimizing tactics through self-play in a simulation or game environment. The term superhuman infers that the AI performs a specific task better than most humans. This scenario poses the question of how humans can be integrated equivalently with an AI application in this decision-making process. Our answer is a combined approach where the knowledge and (control-) responsibility of a human operator (human speed) and the rapid execution of an AI system (machine speed) can work together. The developed human-in-the-loop interface provides the operator with the possibility to observe and change any command given by the AI.

The AI can generate sentences with a fixed vocabulary, which dramatically reduces the complexity of the action space. However, this provides the user with an even more powerful and flexible way to define orders. The so-called strategy AI will be able to command any subsystem, including unmanned aerial vehicles (UAV). Finally, this new approach simplifies human-in-the-loop or human-on-the-loop interactions with trained AI.

While training AI continues to represent a challenge, the concepts, processes, and adequate technologies are available free of charge. In particular, the internet contains various publications with detailed descriptions of the effort undertaken in this field. In particular, "human versus computer" games are commonly used to demonstrate progress in AI development.

2 RELATED WORK

In recent years, AI—and especially deep RL—have been used to solve complex tasks. Since the development of AI systems, testing these systems for usability in a realistic environment has been a vital task. The game industry has a strong interest in using AI in commercial games to increase the number of potential buyers. The more sophisticated and demanding a game is, the more challenging it is to play—especially for a single player. Therefore, games have been a very important test environment to evaluate the performance of AI systems. A great deal of research has been performed to generate superhuman performance in game environments. Mnih et al. (2013) were the first to incorporate deep learning models into reinforcement learning. Using this method, they succeeded in using AI models to play Atari games and handle high-dimensional input. Recent breakthroughs in training AI to play classic board games such as Go (Silver et al. 2016) and complex games such as StarCraft II (Vinyals et al. 2019) have indicated the success of deep RL.

The real-time strategy game StarCraft II is well known and became popular in the RL community through the project AlphaStar from DeepMind, a subsidiary of Alphabet (Google). Notably, this game has many similarities to military combat. In the game, the user or AI must control and craft different units and their capabilities. The game also implements a “fog of war”, which results in the user having limited information about their enemies. The project AlphaStar incorporated the latest deep RL technologies to train AI capable of defeating most human players (Vinyals et al. 2019). The AI architecture and RL setup are used as the baseline for our AI in a military simulation.

In a recent paper, Eloff et al. (2021) trained two RL agents to collaboratively navigate through a 2D maze. The sender agent was tasked with understanding the state of the environment—including the position of the second agent. It was then tasked with creating a movement command (e.g., “Move three blocks north”) using natural language. The receiver agent must then decode the command and perform the action. This approach showed some very promising results and is very similar to the task of this paper. Notably, they had a substantially smaller environment and a more limited action space. The vocabulary in their project consists of 31 English words to create basic instructions (e.g., “Move up three blocks”). In this study, the action space is extended to 125 words with complex implications. Moreover, it was also possible to exactly measure the result and reward of the advising agent for its “strategic” decisions. An interesting finding is the ability of the receiver to correct flaws in the sender's messages and thereby still perform the correct actions.

The publication of Jiang et al. (2019) from Google Research showcased natural language as a communication layer between hierarchical RL agents. They demonstrated the effectiveness of natural

language for long-horizon control problems (Vinyals et al. 2019). Therefore, we used the findings of this paper and adapted them for a larger environment in a military combat simulation.

3 RELEGS – REINFORCEMENT LEARNING FOR COMPLEX COMBAT SITUATIONS

3.1 The "ReLeGSim" Simulation Environment

We reuse the ReLeGSim combat simulation environment developed for an earlier study by the German Armed Forces on AI-based military decision making from 2020 to 2021. ReLeGSim can simulate two opposing forces (attacker and defender). The attacker is tasked with taking a defined area, while the defender attempts to fend off the attack. Both sides command several companies that can contain different platforms. These platforms can be of different types, such as reconnaissance units, battle tanks, or infantry fighting vehicles. Each platform also comes with different capabilities and subcomponents, such as sensors and weapons. Additionally, both sides can lay minefields and request joint fire support in the form of combat helicopters, artillery, or close air support. However, the agents do not possess an omniscient view and thus suffer from the "fog of war", which means that they must actively perform reconnaissance with their units to improve their situational awareness.

To successfully train an RL agent, the agent should be trained in as many different situations as possible. ReLeGSim offers a procedural level generator that enables the generation of randomized scenarios. It is also possible to create levels based on real-world data through Open Street Map. The scenarios contain elevation data generated through algorithmic solutions or derived from real-world models (e.g. Shuttle Radar Topography Mission Data), which can affect the visibility and weapons used in the simulation. To reduce the complexity of the models, the maps are rasterized at a previously defined resolution. The number of units on each side obviously plays a significant role in the outcome of the simulation. Therefore, the composition of units and companies can be specifically defined or randomly generated. The following image (Figure 1) presents a sample grid with blue and red company units.

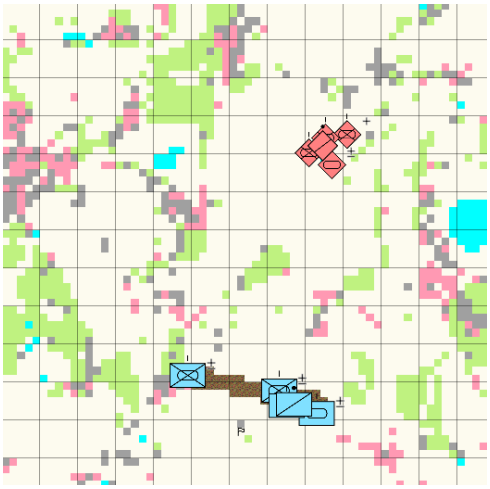

Figure 1: ReLeGs rasterized map with companies.

The simulation core can be executed headless without the graphical user interface and run in parallel for cluster operation and training. To speed up the RL training, ReLeGSim can run faster than real-time (speed ReLeGSim > 600 * real-time). ReLeGSim was developed in the Python programming language and is compatible with the OpenAI Gym standard (Brockman et al. 2021).

With a complete scenario consisting of a terrain model, elevation model, and an initial location of the units, ReLeGSim can be started. As in other turn-based games, inputs from either humans or the RL

agents are made in discrete time steps. Between the time steps, the underlying simulation continuously executes the required actions. When a company is tasked with moving to a specific position, it is continuously moved in that direction until the next discrete time step is reached by the simulation. During the time steps, the RL agents cannot influence the simulation. Shorter time steps provide a more detailed simulation but also require more training resources. Thus, the clock rate has a significant impact on the overall training duration.

3.2 ReLeGs AI Architecture Overview

To gain inspiration for our model architecture, we investigated the architecture of DeepMind's AlphaStar since it is considered state of the art in the field of complex RL problems. With our architecture (illustrated in Figure 2), we propose an adapted and novel approach for a flexible and scalable action space combined with a deep neural network. The design of the observation space is based on military experience in how to prepare a battlefield. It is common to use a map and a table of available troops. As a result, the observation of the simulation is separated into scalar data (e.g., the number of available tanks and their ammunition). Simultaneously, a map-based input is given as visual input to a spatial encoder.

The scalar data is used to advise the AI about almost all details of the scenario. This includes data on its own troops and their platforms as well as partial information about the enemy troops. The inputs are not given in absolute numbers; instead, normalization is used to improve the training performance. The encoder can be easily written as a multi-layer perceptron (MLP); however, the use of a multi-head attention network greatly increases the quality of the trained agents and should thus be used (Vaswani et al. 2017).

To understand the geographic terrain, distances, and meaning of elevation, the AI is fed a visual representation of the map with the entities encoded in it. The color scheme is based on a three-channel image, which allows us to easily visualize the data. Although using more channels would be problematic to graphically display for humans, the AI would be capable of understanding more channels. Different field types and entities are encoded with a special color so that they are always distinguishable. This so-called spatial encoder is composed of multiple convolutional layers. Initially, we attempted to use well-known architectures such as ResNet-50 (He and Zhang 2016) and MobileNetV3 (Howard et al. 2019), even with pre-trained weights. However, this did not lead to acceptable training performance. Therefore, we decreased the size of the convolutional neural network (CNN) with our architecture.

To test and optimize this architecture, we used an autoencoder setup with real samples from the simulation. We were able to decrease the number of parameters from approximately 2 million to approximately 47000. As a bonus, we produced a pre-trained model that was already fitted to the real observations of the simulation. This step helped us enormously in speeding up the RL process.

An optional element is the added language input to define a task for the AI. While this element is not used for the general strategic AI, it is planned for use with the subordinate agents. These agents will receive a task from the strategic AI in natural language and process it using a bidirectional gated recurrent unit (GRU) encoder.

The encoded values of the visual, task, and scalar data are combined and fed into the core network. The core is mainly a long short-term memory (LSTM) component with 768 units, as introduced by Hochreiter and Schmidhuber (1997). In a military scenario, the commander must be aware of the long and strategic planning of high-value assets. In this simulation, the AI can request combat support elements that require up to 15 minutes until they affect the battlefield. Therefore, the AI must understand the timing and planning of tasks for the future. Using an LSTM network in RL is quite difficult since it requires a lot of training time and causes vanishing gradients in the layers above. Thus, we decided to add a skip connection over the LSTM to minimize the negative effects of the added layers.

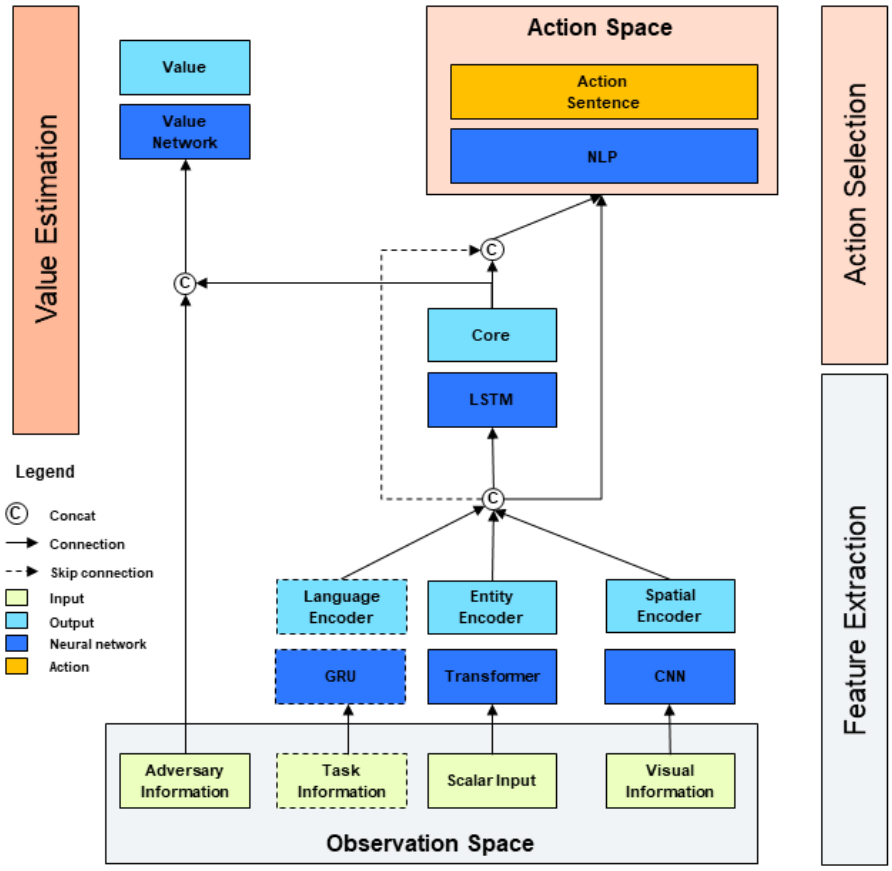

Figure 2: ReLeGs AI model architecture.

The action head consists of a natural language processing (NLP) model. This is a very simplified action head model with a small LSTM and one additional dense layer with approximately 340000 parameters. The result is a multi-discrete action space with dimensions of 8 x 125.

In addition to the main model, a separate part exists for the value network. The value network uses the output of the core LSTM and concatenates the adversary information to pass both into an MLP. The MLP can then precisely predict the value function. Through the adversary information, the value network has a god-like ground truth view of the simulation. Since this network is only relevant for training, it can be performed without interfering with the integrity of the training.

3.3 Action Space with Natural Language

In the first iteration of ReLeGs, we began with a classic multi-discrete action space to command the movements of companies directly. Therefore, we used an action space with 58000 different actions at each discrete time step. This inflexible and complex action space was replaced by a natural language interface.

We now allow the AI to communicate one sentence at each discrete time step. In the first iteration, the AI was only capable of defining the movement of units. Now, our AI has many more ways to distinguish actions. For example, it can tell a company to observe an area, hide in a forest, attack from the left/right edge, and many more actions. This brings the given commands to a different level, from the

micromanagement of movement tasks to more high-level goals. This approach helps the AI to focus on the strategy instead of having to micro-manage every entity.

Although the vocabulary in this scenario remains quite limited, it can easily be extended. The study currently focuses on an attack scenario at the battalion level. Therefore, it is limited to these doctrinal terms and can be extended to a broader range of scenarios and terms. We allow the AI to use a vocabulary of 125 words. These words can be freely used in sentences with up to eight words. This limitation is added to reduce the complexity and use a multi-discrete action space. Therefore, the AI could command the following:

"`Company 2_110 and Company 2_115 attack left comprehensive Enemy 1`".

To improve the learning performance, words that are invalid in the given context (e.g., the names of destroyed units) are masked out. The masking helps the AI to only predict valid operational commands. The newly proposed action space improves the readability and explainability of the commands given by the AI.

3.4 Reward

The setup of a reward function in RL is a fundamental and challenging aspect. The design must be adapted to the simulation and the expected behavior of the agents. The training does not consider tactical-looking movements or other military doctrines. In essence, the agent's goal is to maximize the cumulative reward (Sutton and Barto 1998), not the immediate reward at each time step. As learning progresses, the agent learns to maximize the reward.

As a result, projects such as AlphaStar created AI agents that perform micromanagement to obtain the absolute maximum reward. These results do not transfer well to a military perspective. For our use case, we do not need the best strategy within the simulation, but a strategy that could be transferred to real life. Another important aspect is the granularity of the reward. If we define a granular reward that is too fine, we will force the agent to behave as we expect it to. As a result, the agent will never be able to explore and find new strategies.

In this special context, we must define multiple reward functions. On the one hand, there is the strategy AI, which will be rewarded based on the battle outcome. On the other hand, we have subordinate agents who execute tasks given by the strategy AI. For this, we propose the generation of a reward function based on the given task. For the subordinates, the final goal of winning the battle is not applicable. Therefore, we must define the rewards based on specific tasks. For example, there should be a reward scheme for a reconnaissance task. The agent executing this task will be judged on how well the specified region is observed to detect enemies without being attacked. For an attack task, the reward function looks completely different. In this case, an attacker focuses on taking the ground and eliminating any enemies as quickly as possible. Thus, the time component, battle damages, and goals in the reward are entirely different.

4 EXPERIMENTS & RESULTS

4.1 Experiment Setup

Many RL experiments begin with a supervised learning phase to incorporate the first agent behaviors without requiring too much computation time to explore the world. Thus, RL is only used to further improve the learned strategies. Due to the nature of our simulation, we do not have any user-played replays. Although creating replays from scripted behavior did improve the initial learning, it fixed the agent on certain behaviors and caused interference with the free RL. Thus, we decided to move in another direction. Since we can fully control the simulation, we were able to set up a very steep learning

curriculum. In this approach, we teach the agents the expected behavior with easy scenarios and then continue to increase the complexity step by step.

Since we are not focusing on theoretical research, we used state-of-the-art algorithms and open-source implementations to build up our environment. ReLeGs mainly use the RLlib and TensorFlow framework. Both frameworks have been proven to perform very well. For optimizing the policy of the agents, we focused on policy-based algorithms such as asynchronous proximal policy optimization (APPO).

Working closely in a military environment with the German armed forces pushed us to use a private computing cluster. Therefore, we did not have unlimited resources to train the AI models. In this study, we used off-the-shelf commercial hardware such as the Nvidia RTX 3090 GPU.

4.2 Pre-Training

The natural language-based action space carries a lot of complexity in itself. Thus, the sentences must be valid and have significance. To cope with this added complexity, we proposed the training of a new action head to its full extent before starting the simulation-based RL. We aimed to achieve an action head with frozen weights that can predict only effective sentences. To achieve this goal, we proposed a two-step training process. As an initial pre-training, we use an autoencoder with a rule-based sentence generator. The sentence generator predicts all possible and valid sentences as an input for the training.

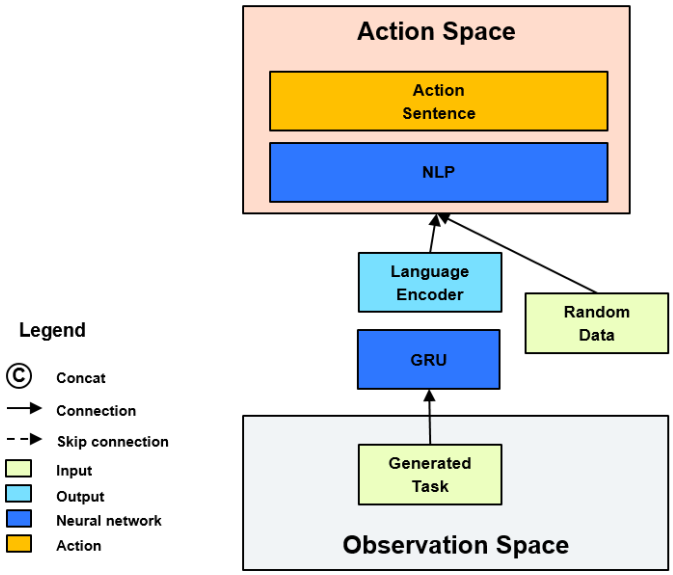

Figure 3: Pre-training.

In this phase, we train the model to be able to encode and decode any sentence. However, when we extract the decoder (sentence generation part) and feed it with random data, it does not predict correct sentences. If we were to use this model directly in the simulation-based RL, we would still need to continue training the model to improve the sentences. Since we aimed to focus on the tactics and not the validation of the predicted sentences in the simulation-based RL, we proposed enforcing the correctness of the sentences by continuing the training of the action head model with random data. To achieve this, we use RL-based training, which uses two different types of inputs (illustrated in Figure 3).

On the one hand, it uses the latent space from the automatically generated sentences with the pre-trained encoder. In this case, we know the expected outcome and reward the model for predicting it

correctly. On the other hand, we use plain random data as an input. In this case, we can only validate the correctness of the sentence. Therefore, we designed a reward scheme to support both kinds of inputs. The design of this reward scheme is fragile and must be balanced flawlessly between both inputs. Figure 4 presents an imbalanced reward design that caused invalid training at later stages. Notably, this is a very extreme result. Most often, we only observe the exploitation of rewards; however, in this case, the exploitation led to poor results for 50% of the samples (samples with known inputs).

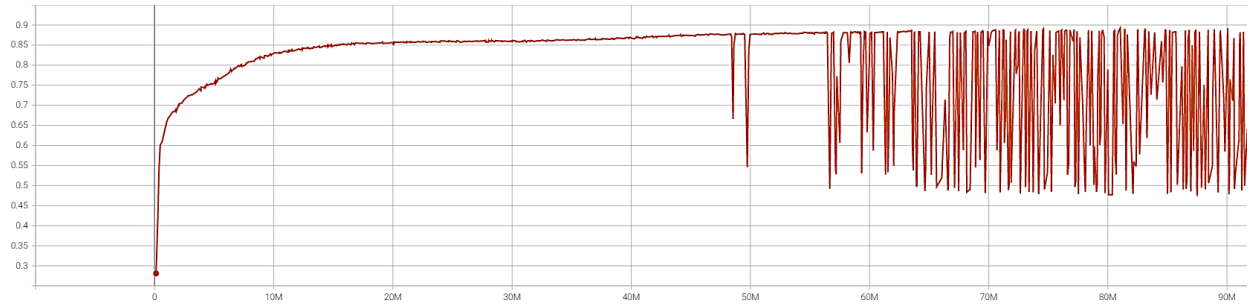

Figure 4: Pre-training with an invalid reward function.

After balancing the rewards, we were able to showcase that this approach works and can be trained in a short amount of time. Figure 5 shows the improved training, which was performed in 2 hours. The final model can almost always predict valid sentences.

Finally, the used reward counts the existing mistakes in a sentence. While using predefined input sentences, any deviation from the expected result is an error. Regarding the random data, every grammatical error is counted word-wise. The reward is calculated as follows: *1.0 – (mistakes / 10.0)* and clipped at zero. As a result of the training, we were able to reach a reward of 0.9 with the mixed input data.

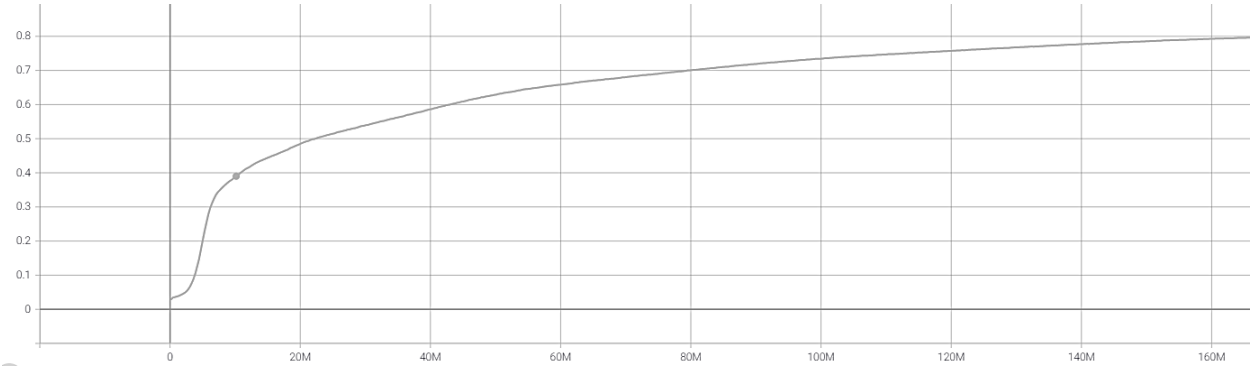

Figure 5: Pre-training with the final reward function.

4.3 Experiments Concerning Architectural Aspects

Our mission is to build a military decision support tool for battalion commanders. Therefore, we are exploring different techniques to make AI capabilities useful for military commanders. As one step toward this goal, we designed an interactive dashboard to explore the agent's behavior. The dashboard allows us to visualize win rates, lost entries, predicted movements, and combat locations between opposing forces. We can select a battleground, create the relevant maps automatically using OpenStreetMap, and place our entities. To analyze the behavior, data farming as described in Horne and Meyer (2004) with thousands of runs is used in the background. On one hand, we evaluate the statistical values (e.g., win rate) depending on whether we provide the defender with joint fire elements such as

combat helicopters. On the other hand, we visualize the predicted combat locations and enemy course of action on a map (Figure 6).

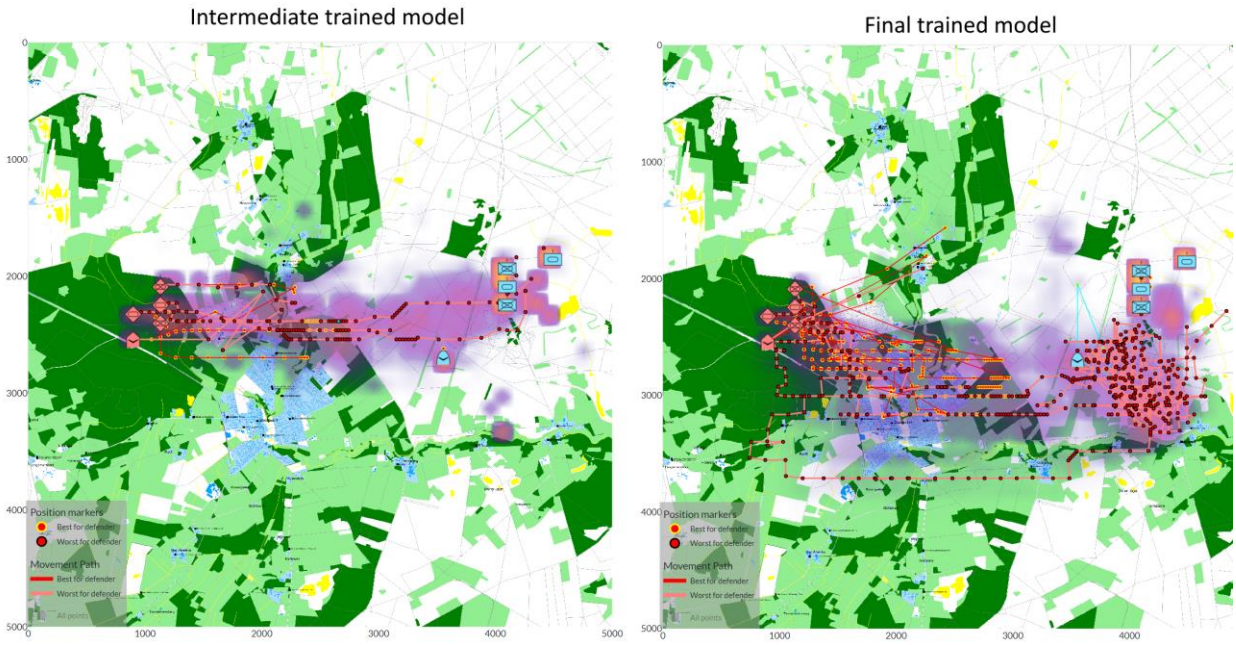

Figure 6: ReLeGs analysis with a map in the dashboard.

In the development phase, we use the same tool to evaluate our AI models and visualize the trained behaviors. For instance, we can compare the strategies of an intermediate trained model with our final trained model. As shown on the left side of Figure 6, the intermediate model already forms a strong attack but does not deviate much from a frontal attack. Instead, the final model learned to attack from the south or the north to improve its win probability.

4.4 Experiments Concerning Natural Language Aspects

Another step toward a military decision support tool is the introduction of the natural language interface to increase the understandability of the agent's decisions. Since this represents ongoing research, we can only share our initial findings and experiments regarding this new action space.

The use of the pre-trained model seems promising and mitigates issues linked to incorrect sentences. However, it also adds a new dispute with the exploration-based training. We had to test and compare the theory that the fully trained action head from the two-phase pre-training is useful for simulation-based training. Instead, we recognized a major concern with this model: it was already fixed on generating correct sentences but was unable to adapt the generated sentences as needed. Therefore, we decided to integrate the second phase of the pre-training into the simulation-based training and started from the first phase pre-trained model. Figure 7 presents a comparison of the old action space and the natural language-based action space.

The AI model with the language-based action space is faster in learning how to reach the initial goals, taking approximately 38% of the samples to reach a similar result. Since the subordinate behavior is still implemented algorithmically, we cannot fully compare it against the micromanaging AI model.

Surprisingly, the difference between the pre-trained action head from phase one and the randomly initialized action head is minimal. The reason for this seems to be the need to explore and train the majority part of the AI model. Ultimately, while the pre-training phase is not required, it helps to develop

and further improve the AI model architecture and reward function. As previously mentioned, designing the reward function is complex and critical for the success of the model.

The other important topic is the human-in-the-loop interface. The given commands from AI can be observed, evaluated and in real time overwritten by the operator. This brings human interactions to the core of the application and fosters trust in the AI system.

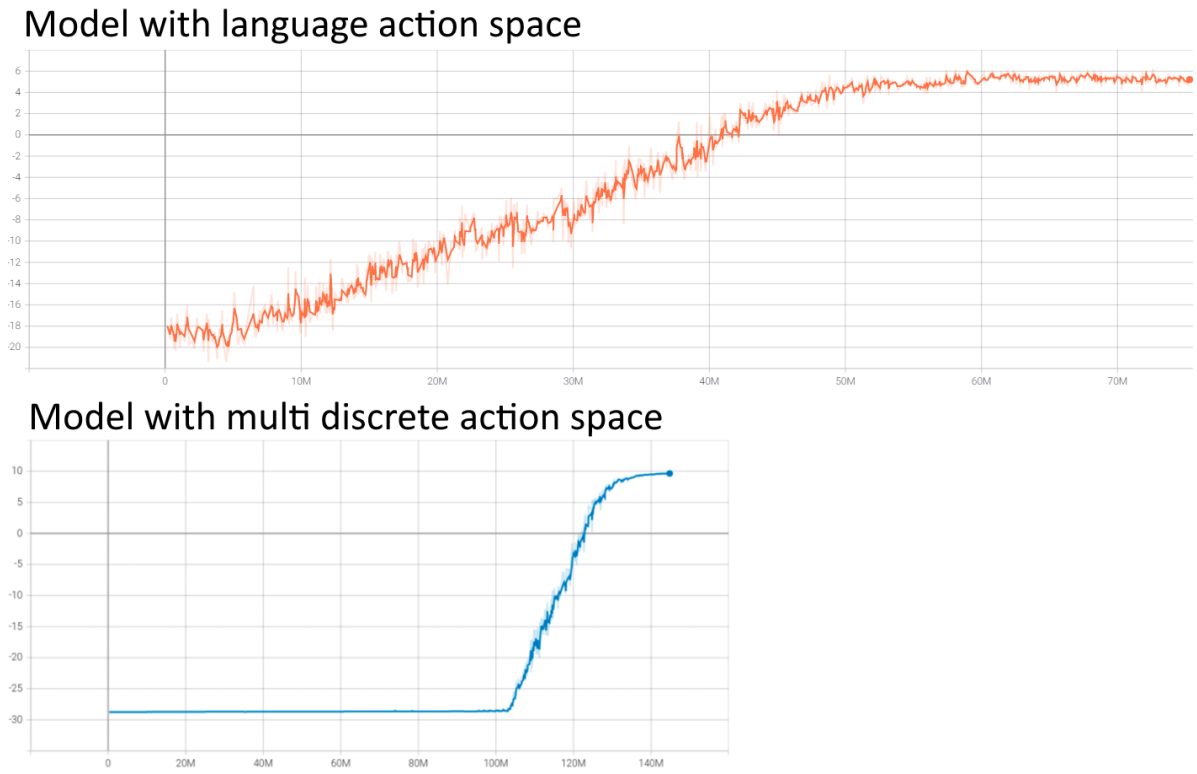

Figure 7: Action space comparison (language vs. multi-discrete action space).

5 CONCLUSION

This paper presents a novel approach to military decision support. It allows us to simulate and analyze a given scenario to assess and evaluate our own strategies against AI forces. Simultaneously, the goal is to integrate an understandable interface for the AI forces to allow a comprehensive understanding of the AI's decisions. The aggregated AI decisions are easier to understand and allow a human to judge the given commands. At the current development stage, this is still active research and not a full product. However, we already see big potential for a future product to be used in real military environments. The language-based action space is easy to understand and can be extended to support different needs.

The decision support tool is successfully designed and implemented through an interactive application and dashboard. It has been shown that this application significantly improves the ability to understand agent strategies and the performance of various combinations of capabilities. This process has helped identify weaknesses and possible areas of improvement for the simulation model by visualizing agent behavior that was not previously observed or understood.

6 THE WAY FORWARD

Research on natural language-based military decision support is still ongoing and far from reaching its limits. While we are currently working with scripted subordinates, we want to introduce hierarchical multi-agent training to increase the capabilities of the agents. This will push the strategies to the next level.

The other open topic is the limited vocabulary. In future iterations, the vocabulary should be increased to enable more flexibility in the command structure. This imposes new challenges with scripted behavior, rewards, and the planned hierarchical multi-agent training.

Later this year, we will bring the research into a live experiment with real UAVs. For this experiment, we will use multiple drones as subordinate entities controlled by the strategy AI with a focus on autonomous reconnaissance.

ACKNOWLEDGMENTS

We would like to thank the entire ReLeGs and KITU study group—and especially LTC Jan Wilhelm Brendecke and MAJ Stefan Göricke from the Army Concepts and Capabilities Development Center and LTC Silvio Püschel from the German Army Headquarters —for their highly valuable contributions to our studies. We would also like to thank Matthias Flock, Matthias Behm, and Sebastian Steinmann from Airbus for their invaluable support during the ReLeGSim and RL implementation.

REFERENCES

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba. 2016. „Openai Gym“. *ArXiv Preprint* ArXiv:1606.01540.

Eloff, K. and H.A. Engelbrecht. 2021. "Toward Collaborative Reinforcement Learning Agents that Communicate Through Text-Based Natural Language". In Proceedings of the *Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA), January 27th-29th*, Potchefstroom, South Africa, 1–6.

He, K., X. Zhang, S. Ren and J. Sun. 2016. "Deep Residual Learning for Image Recognition", *2016 IEEE Computer Vision and Pattern Recognition (CVPR)*, 770-778.

Hochreiter, S. and J. Schmidhuber. 1997. "Long Short-Term Memory". *Neural Computation 9(8)*, 1735–1780.

Horne, G. and T. Meyer. 2004. "Data Farming: Discovering Surprises". In Proceedings of the *2004 Winter Simulation Conference*, edited by R. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 171–180.

Howard, A., M. Sandler, G. Chu, L.C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q.V Le and H. Adam. 2019. "Searching for mobilenetv3". In Proceedings of the *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1314–1324.

Jiang, Y., S. Gu, K. Murphy, and F. Chelsea. 2019. "Language as an Abstraction for Hierarchical Deep Reinforcement Learning". In *Advances in Neural Information Processing Systems 32*, 9414-9426. Red Hook, Curran Associates, Inc.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller. 2013. "Playing Atari with Deep Reinforcement Learning". https://arxiv.org/abs/1312.5602, accessed 8th June, 2022.

Silver, D., A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 2016. “Mastering the Game of Go with Deep Neural Networks and Tree Search”, *Nature* 529(7587), 484–489.

Sutton, R. S. and A. G. Barto. 1998. “Reinforcement Learning: An Introduction”. *MIT Press*, Cambridge, 1998, vol. 1, no. 1.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin. 2017. “Attention is All You Need”. http://arxiv.org/abs/1706.03762, accessed 8th June, 2022.

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps and D Silver. 2019. “Grandmaster Level In StarCraft II Using Multi-Agent Reinforcement Learning”. *Nature* 575(7782), 350–354.

AUTHOR BIOGRAPHIES

MICHAEL MÖBIUS is a senior software and AI engineer working in the department “Operational Analysis and Studies” at Airbus Defence and Space in Germany as a technical lead for simulation and AI related studies and projects. He also teaches at the Baden-Württemberg Cooperative State University in Germany to share his software engineering knowledge with young students. He started developing software at 10 years old and sold his first commercial software at 16. As of today, his main research interests include stochastic simulation design and analysis, autonomous systems, reinforcement learning, and operational analysis. His email address is michael.moebius@airbus.com.

DANIEL KALLFASS is an expert in operational analysis in the department "Operational Analysis and Studies" at Airbus Defence and Space in Germany. He has more than 17 years of experience in leading national and international defense and security related studies and research projects in the field of simulation-based operational analysis, distributed simulations, and artificial intelligence. His main focus is on the development and application of simulations such as the 3D agent-based simulation PAXSEM in combination with data farming and the latest artificial and deep learning techniques, such as deep reinforcement learning simulations. His email address is daniel.kallfass@airbus.com.

LTC THOMAS DOLL is head of the department of Army Concepts and Capabilities Development Center, where he is responsible for conducting military analyses and studies. His main areas of interest are modeling and simulation, the development and deployment of unmanned systems, and artificial intelligence. From 1990 to 1994, he studied electrical engineering at the University of the German Armed Forces in Munich. He received his Master of Science degree in 2004 from the Naval Postgraduate School in Monterey, California, USA. His email address is thomasmanfreddoll@bundeswehr.org.

LTC DR. DIETMAR KUNDE is a German Army officer who completed the Modeling, Virtual Environments, and Simulation (MOVES) Ph.D. program at the MOVES Institute of the Naval Postgraduate School (NPS) in Monterey, California in December 2005. He received his M.S. in Surveying from the German Armed Forces University in Munich in 1985. In 1997, he received his M.S. in Operations Research at the NPS. His research covers military decision support, human performance modeling, cognitive modeling, and artificial intelligence. For the past 10 years, he has been working as a branch head for operations research and modeling and simulation for the German Army Headquarters. His email address is dietmarkunde@bundeswehr.org.