Transition from Rule-based Behaviour Models to Learning-based Behaviour Models in Tactical Simulation Environments: A Case Study

Arif Furkan Mendi*, Fatiha Nur Büyükoflaz, Tolga Erol, Dilara Doğan, Turan Topaloğlu, Şenol Lokman Aldanmaz, Mehmet Kozan
Simulation, Autonomous and Platform Management Technologies
& Command, Control and Defense Technologies
TURKEY

{afmendi, fbuyukoflaz, terol, ddogan, ttopaloglu, saldanmaz, mkozan}@havelsan.com.tr

H. Oktay Altun, M. Esat Kalfaoğlu
AutoDidactic Technologies A.Ş.
TURKEY

{oktay.altun, esat.kalfaoglu}@autodidactic.com.tr

ABSTRACT

Forces in virtual environment (FIVE) simulator software developed by HAVELSAN provide a comprehensive tactical and operational training environment in a safe and cost-effective way, utilizing various virtual warfare equipment (such as weapons, sensors, and communication tools etc). Currently, behaviour models managing FIVE entities is highly dependent on the rule-based behaviour developed by field experts and system engineers. However, the rule-based operation of FIVE software requires intensive programming and field experts' guidance, and hence highly labour intensive. Furthermore, complexity and burden of this task increases significantly with the complexity of the scenario. In addition, the virtual entities with rule-based behaviour have standard and predictable reactions to their environments. Therefore, in this study we present the transition studies from rule-based behaviour to learning-based adaptive behaviour via reinforcement learning techniques coupled with other machine learning techniques, namely FIVE-ML project. For this aim, primarily reinforcement learning based behaviour models are trained for both air-to-air and air-to-ground scenarios up to six virtual entities. It is observed that virtual entities trained with reinforcement learning dominates existing rule-based behaviour models. During these experiments, it is also observed that utilizing supervised learning as a starting point before reinforcement learning reduces the training time significantly and creates more realistic behaviour models.

1.0 INTRODUCTION

Today, training pilots who will use aircrafts is of foremost importance. Training pilots with real aircrafts is quite difficult for reasons such as airspace regulations, excessive costs and risks that may occur during training, and complexity of creating a real-world scenario including realistic defence and warfare platforms used by adversaries or allies. Flight simulations used in pilot training often work integrated into tactical environment simulations. Through these tactical environment simulations, pilots are trained via controlling high-fidelity airplane models in the presence of many low-fidelity entities completing the scenario. These low-fidelity assets are created and controlled by a computer and often named as Computer Generated Forces (CGF) [1], which are autonomous units representing defence or attack systems on air, land or sea surfaces.

CGFs are used in the preparation process for the deployment of personnel, in tactical training, or in the development of new strategies. CGFs need different programming for each application (or each scenario). These forces, created by traditional methods, cause non-adaptive and inflexible patterns of behaviour. This

leads to students receiving simulation training in the presence of statically programmed assets and reducing the quality of their training. When new scenarios are necessary, specialists are needed to create new scenarios. In addition, since scenario creation will be performed using classic control branches, it is often infeasible to consider all possibilities in the process of creating a new scenario, even possible, it is a quite challenging task. For these reasons, there has been a growing need for more realistic virtual environments and new scenarios to adapt to the ever-changing world to simulate both the pilot candidates own missions and the current capabilities and tactics of opposing forces.

In this study, a transition towards artificial intelligence-oriented behaviour modelling rather than traditional scenario-specific modelling was proposed as a solution to the problems described earlier. In other words, virtual entities will be transformed into dynamic virtual entities that can learn. But there are many situations those virtual entities need to consider in their training processes. First, they must learn to respond appropriately to the environmental factors they perceive with their sensors. It must then recognize his friends and foes and act in accordance with their class information and the types of munitions that attached on them. It should be able to act with team actions in cooperation with his friends.

The preferred method of machine learning for adding intelligence to virtual assets is reinforcement learning (RL) [2] for a fundamental reason: the actions that the entities will take has delayed consequences. In recent years, RL has been recognized as a new way to solve complex and unpredictable control problems compared to traditional control methods and utilized in many fields such as robotics, computer vision, autonomous driving, advertising, medicine and healthcare, chemistry, games, and natural language processing [3]–[9]. The studies in the literature have boosted since the introduction of deep learning to RL concepts (i.e., deep RL [10]) as in the case of many challenging computer vision and natural language processing tasks [11]–[15]

For this aim, in this study (i.e., FIVE-ML), the first stage experiments of the transition from the rule-based behaviour model of FIVE software of HAVELSAN to the RL-based behaviour models have been implemented. From these experiments, it is shown that intelligent virtual entities trained with RL algorithms outperform the currently existing rule-based entities of HAVELSAN in both air-to-air and air-to-ground scenarios. Moreover, the joint implementation of imitation learning [16], [17] and RL has also performed successfully, which has fastened complete transition process of FIVE software.

It is envisaged that a simulation that develops new strategies by learning from the choices made by pilot candidates will take pilot training to a much different point. When the project is completed, a new system will be designed that will allow the training of more equipped and specialized fighter pilots in their fields. An existing rule-based scenario system will evolve into a system that can update itself. Thus, pilot candidates will have the opportunity to develop ideas against new strategies discovered by intelligent entities instead of being content with the knowledge and experience of the experts in the field. In addition, from a scenario mechanism that has been prepared with a lot of effort, the computational scenario automation mechanism will automate the entire process.

2.0 RELATED WORK

Related work in the literature is given under the two main categories. These are learning-based behaviour modelling in flight simulation environments and actor-critic networks in deep reinforcement learning.

2.1 Learning-based Behaviour Modelling in Flight Simulation Environments

Rapid advances in computing hardware technologies and advances in RL algorithms have brought up the possibility that virtual entities in tactical simulators can gain intelligence with the experience gained from interacting with the environment. In traditional simulation technologies, the behaviour of virtual entities is governed by scripts that anticipate predetermined actions on a particular scenario. The preparation of rule-

based scripts brings along problems such as high complexity, rigid and unrealistic behaviour. Numerous studies have been conducted in the literature to overcome these problems. In this section, the previous studies will be examined within the relation to FIVE-ML.

Especially in combat aircraft training simulators, one-to-one air to air combat, also known as dogfighting, is one of the crucial missions in pilot training. In this regard, the more realistic the battle between the human-controlled simulator and the computer-controlled virtual plane, the more the training mission will be achieved. The study, which can be regarded as the first of the studies on this subject, was carried out in 2012 [18] indicating that a computer-controlled virtual airplane with adaptable characteristics can be more effective than a rule-based computer-controlled virtual airplane for simulator-based training of fighter pilots. The learning based virtual entity is guided by a self-organizing neural network called FALCON. Based on the neural network which uses adaptive resonance theory (ART), one can learn and generalize situations in stages. It can also discover information during real-time interactions with the environment using RL. The study is collaborated with a multinational simulator manufacturer to evaluate the performance of smart virtual assets. An experiment consisting of two groups of subjects (two trainee pilots and three experienced pilots) was conducted using a commercial class simulation platform. Looking at the initial difficulties that the intelligent virtual asset presents to trainee pilots and experienced pilots, it has been shown that it is more suitable for modelling the opponent with unexpected manoeuvres than a rule-based virtual entity. On the other hand, it has been noticed by experienced pilots that smart virtual assets do not adapt well to missile war scenarios.

In the report prepared by J. Roessingh et al. in 2011, with the aim of developing simulation training for fighter pilots, how virtual forces that display realistic tactical behaviour can be synthesized with artificial intelligence techniques are discussed within the scope of the smart bandits project. It was concluded that there was no single technique to solve all the tactical problems of intelligent CGFs involved in an air-to-air mission, and a hybrid system was proposed. Since the desired responses in high complexity environments such as simulation are usually unknown, RL is preferred to improve the responses in each experiment, while the neural network is used to approach the state-action space that occurs during RL [19].

In another article published in 2015 by Roessingh, A. Toubman et al., it is discussed that CGFs are problematic from the perspective that they were inappropriately rewarded and punished in some cases because the missile hits were stochastic. It was thought the reward design should take this problem into consideration and they propose a new reward design to overcome this issue. Tests show that the use of this new function, which rewards virtual assets based on the expected outcome of their actions, significantly improves the performance of CGFs in various scenarios compared to the previous reward function [20].

Another of the studies conducted to make simulation training for warplanes more effective was published by J. Källström and his friend in 2019. This study discusses how machine learning techniques can be used to automate the process of creating advanced, adaptive behavioral models for constructive simulations. As a result of the first experiments conducted many times, it has been shown that RL, especially multi-agent multi-objective deep RL, enables synthetic pilots to learn to cooperate and prioritize between conflicting targets in air combat scenarios. Although the results are promising, it has been concluded that further algorithm development is required to fully master the complex field of air combat simulation [21].

In the continuation study published by J. Källström et al in 2020, the applications of multi-agent deep RL that learns coordination in air combat simulation were examined. Three main use cases have been determined for RL within the scope of the application area. In the first scenario, curriculum-based learning was used to help virtual assets learn against sparse rewards and wide search space problem. In the second scenario, multi-agent deep deterministic policy gradient (MADDPG) and deep deterministic policy gradient (DDPG) methods are used. Curriculum learning has been shown to be a promising approach to address the complexity of the air battlefield, while multi-objective learning is a promising approach for creating virtual entities with a variety of attributes whose behaviours can be adjusted after learning has been completed [22].

A study was conducted in 2019 on improving the control system of flight simulators using RL. The flight simulation control with two degree of freedom is studied and three model free RL algorithms are examined for this aim which are DQN, DDPG and PG. As a result of the analysis of the methods used, the DQN algorithm has been proven to be more suitable for discrete tasks than the other two methods. Combining the advantages of PG and DQN, the DDPG algorithm has been shown to perform very well in complex tasks. It has also been observed that in order for model-free RL algorithms to obtain sufficient data, the virtual entity must constantly explore the environment until the system reaches a state of convergence. Accordingly, it is emphasized that the inefficiency of the data is an inevitable end, and increasing the efficiency of the algorithm is the primary problem that needs to be solved [23].

In the master thesis published in 2018 [24], how RL algorithms can change the dogfight scenario in air-to-air simulations discussed. Tactical simulation environement (TACSI) produced by Saab AB was used in the study. A genetic algorithm (GA), one of the RL algorithms for the relevant scenario and genetic programming (GP) has been used as an optimization algorithm to investigate the solution field in the simulation environment and to develop artificial intelligence behaviours. Behaviuor tree (BT) by combination of genetic programming techniques is presented. As a result of the training, it was observed that the BT framework utilized a lot of computational power to constantly check conditions and pass the tree for each confirmation. To change this, it has been predicted that memory nodes that reduce the repetition of tree transitions instead of BT may be more effective [24].

Smart bandits, which is started under the roof of the Royal Netherlands Space Centre (NLR) is a research project created with the contribution of a large number of doctoral students looking for improved methods for behaviour modelling. A study published in 2018 as scope of the project stated that the behavioural capabilities of CGFs created by traditional modelling techniques are quite basic. For this reason, behaviour modelling has emphasized the need for a paradigm shift of CGFs towards agent-based modelling rather than traditional scenario-based modelling in order to created robustness against the dynamic environments. In this context, a learning-based artificial intelligence solution for modelling the behaviour of simulated forces is presented [25].

A dissertation study was conducted in 2020 to explore how RL can be used in a flight simulator with the goal of improving synthetic enemies. In this study, the aim is to provide the virtual entity with a behaviour model which has an ability to defeat a computer-controlled synthetic enemy in TACSI simulator environment thanks to Q-learning implementation. As a result of the study, it was said that existing approaches of Q-learning in the field of RL are promising and give surprising results in some applications. But due to the lack of training time required to train the behaviour of artificial intelligence and some limited working conditions, it was concluded that the trained virtual entity could not perform as good as the behaviour performed by an expert [26].

In the literature, there are also some recent works that focuses on the intelligent manoeuvre model of unmanned aerial vehicles (UAV) from the control theory perspective which is crucial for future autonomous aerial combats [27], [28]. Manoeuvre model is a continuous domain problem and it is seen that DDPG [29] is chosen for this problem in these studies. In [27], some adversarial perturbations are given in order to provide agents robustness as in the real world. Additionally, a reward shaping method based on maximum entropy (MaxEnt) inverse reinforcement learning algorithm (IRL) is proposed to improve the efficiency of the algorithm.

2.2 Actor Critic Networks in RL

Reinforcement learning is a machine learning technique that tries to maximize the total reward obtained at the end of the episode. Contrary to supervised learning, there is not a state action pair that can be learnt. Instead, the aim is to learn an action for a given state to maximize the total reward at the end. In the application of RL, there are two main elements which are agent and environment. Agent takes an action for a

given state in the environment and observes the reward of this state action pair. The brain of an agent is called policy function which maps the state to an action.

Actor-critic networks has become one of the most common terminology in deep RL algorithms. In this terminology, actor represents the policy function and critic represents the value function. Value function represents the expected value of the discounted sum of the rewards for a given policy. For the policy part, there is not significant modifications between different algorithms. However, the utilization concept and benefits of critic may vary in different deep RL algorithms.

One of the way that critic can be utilized is to predict the value function so that the update is implemented without waiting the end of the episode [30]–[32], which is called bootstrapping. By this way, the update frequency is increased which is beneficial to make the training faster. With this implementation, update is possible even with a single action step in the environment. Another advantage of critic is that its output can be utilized directly to in the loss function for maximization purpose which is parallel to maximizing the reward [29], [31], [33] . This is an alternative to policy optimization methods, and this change enables the utilization of off-policy algorithms which also utilizes the old experiences before the network update, increasing the number of experiences and diversity for the update loop. Thirdly, it can be utilized to reduce the variance of cumulative reward (rewards to go) in policy optimization methods, which is indicated as good practice for RL [32], [34].

Most of the scenarios in virtual environments contains more than a one single agent since generally two opposing teams compete. Based on the architecture which will be compatible with the simulation environment, there are mainly three approaches to implement actor-critic algorithms on virtual assets. Most preferred of the approaches is centralized method which can be addressed as god-view mode. In this method, global merged information gathered from all assets is fed to actor and critic networks as a state input. Hence, all the agents contribute their experiences during episodes to central actor and critic networks which are used by all the agents commonly. An important benefit of this approach is an ability to enabling cooperative relationships among the agents since the policy has an awareness of all different agents and their roles in environment. However, this sort of approach may not be realistic and can damage realism of the virtual environment due to limited awareness of assets in real world cases. The other approach is called decentralized learning and based on independency of the agents. In this method, each individual agent in the environment has its own actor and critic policy and each agent uses its own local observations. The situation becomes more real world likely in this approach since agents can partially observe and interacts with the other assets. Decentralized method relies on converging to specific behaviour models of each individual agents because of their private policies. Nonetheless, decentralized methods usually can lead agents to unstable policies due to the uncertainty arising from the other agent's unpredictable actions on the environment. The last one of the approaches combines the first two approaches and varies according to implementation preferences. Common usage of combined method uses a centralized critic and decentralized actor policies. Agents tend to learn specific roles using their local actor policies as in the real world. Central critic network helps to create a common sense of motivation among all agents. In addition, common value function feeds on all agents, providing a broader perspective for agents to evaluate their local actions. The important implementation of this concept is MADDPG algorithm [35].

3.0 TACTICAL SIMULATION ENVIRONMENT

Simulation environment we utilized in this project is a commercial tactical environment simulation software and we reason the assets allocated within it with RL algorithms. This software is a simulation software produced to ensure that tactical training can be delivered effectively at an affordable cost by developing various scenarios for combat platform simulators. The behaviour of air, land and sea platforms can be virtually modelled to include various combat equipment, thus providing the necessary tactical environments

for simulators.

The general structure of the software designed using distributed architecture is given as follows. Model editing allows editing the attributes of the physical models for assets in the simulator environment and the subsystems or munitions that these assets can use. In the model editing module, the first of these subsystems, different systems in the tactical environment software inventory (e.g., tanks, fighters, and submarines, etc.), equipment used by these systems (e.g., weapons, sensors, jammers and communication devices, etc.) and the behaviour patterns that systems will perform in different scenarios are modelled. In the scenario editing module, which is the second module, the existing rules and the behaviour of the systems in the scenario are related in order to create the virtual operation environment required in accordance with the training requirements.

Scenario editing interface allows adding assets to the scenario to be run, changing asset types, positioning, loading behaviour rules for assets, and assigning routes. All kinds of factors and details regarding the fiction of the scenario are arranged in this section. The simulation execution module is the module on which the prepared scenarios are executed. In a module consisting of five subunits in itself, the main architecture allows all subunits to run synchronously in real time. The HLA interface serves as the module's external interface unit. The rule engine interprets the codes defined by the Python language, performs the action and performs the behaviour that the entity will display in the next stage (for example, fire, run, and attack, etc.) determines. This is the unit that will be modified the scope of the project. This rule-based unit will be transformed into an artificial intelligence-based system. The other three subunits perform physical simulation of the entities within the scenario.

4.0 METHODOLOGY

In this part, the original PPO is explained and then two main algorithmic PPO concept which are utilized in this study are explained. These are hybrid PPO and Four-Worker PPO algorithms. In the tactical environment of FIVE simulators, there are multiple action groups that needs to be handled. These actions can be categorized under the two categories which are manoeuvre actions and discrete tactical actions. There are three main manoeuvre actions which are to rate of change of speed, heading and altitude. Manoeuvre actions are responsible for controlling the aircrafts' three-dimensional movement in the environment. Manoeuvre actions have different implementation methods where the actions are considered as continuous scalar quantities or discretized into subsets along the range of possible manoeuvre at unit frame time.

The discrete tactical actions can also be divided into three subsets which are self-actions, entity related fire-sensor actions and entity related message actions. Self-actions are related with only the controlled entity such as turning on/off sensor, activate/deactivate route, turning on/off jammer actions. Self-actions do not require to interact with other assets in the environment but they cause an effect on the other entities including missiles. These actions have an importance, they assist the agent by allowing to agent to control its own aircraft and providing environmental information about situation, showing the big picture to the agent. Most of the time, timing of self-actions can be quite critical since some of them such as turn on/off radar action enables applying entity related actions such as fire guided missile to potential threat. Entity related fire-sensor actions includes both self-entity and target entity such as firing a missile or tracking and designating entity.

Entity related actions provide activity to the agent by creating a directly aimed effect on the other assets which are actually interesting according to the agent. Entity related message actions are related with sending a message to target entity by using radio communication. Because the action space that needs to be handled is a complex space, the original PPO implementation is not directly utilizable. Therefore, hybrid PPO implementation and four-worker PPO implementation are adopted in FIVE-ML.

4.1 PPO

PPO[32] is a deep RL algorithm and utilizes a policy optimization method which takes its roots from the RL algorithm. It is working in an on-policy manner such that only the experiences after last update are utilized in the next model update. It has an implementation for both continuous and discrete action spaces. For discrete action space, categorical sampling is utilized in order to implement exploration, while Gaussian sampling is chosen for exploration in continuous action space.

Policy optimization methods requires the update within the specific region which is called trust region [34] because of the first order approximation in deriving the method. Therefore, TRPO implements it as KL-divergence constraint, while PPO implements the clipping mechanism. The clipping objective is called as surrogate objective and can be written as below:

$$L(\mathrm{s},\mathrm{a},\theta\mathrm{k},\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta k}(a|s)} A^{\pi_{\theta_k}}(s,a),\ \ clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right)$$

in which ϵ is a (small) hyperparameter which roughly says how far away the new policy is allowed to go from the old policy. Due to the unstable characteristic of the environment, this value is usually kept small and it allows policy to slightly change from its previous baseline. Although clipping concept is claimed to superior over KL-divergence concept implementation, there are also works which claims that the superiority of PPO over TRPO is not mainly because of clipping mechanism but because of the code-level optimizations [36].

4.2 Hybrid PPO

Hybrid PPO treats the action space as two groups in order to cover entire possible actions which can be applied by the assets in the environment. These are manoeuvre actions and tactical discrete actions. For this aim, two policy function and a single central critic function is utilized. Critic network uses state input to calculate value of the actions. Manoeuvre actor policy estimates the rate of change speed, heading and altitude to be applied for movement of the agent in the following step while tactical actor decides to which tactical action will be chosen and treats all the tactical actions as separate output nodes. Tactical actor policy basically calculates the probability distribution among the tactical actions according to return of reward function. Under mentorship of critic, tactical actor tries to popularize actions with good outcome and suppress actions with bad outcome. The surrogate objective losses of two parts are also aggregated.

4.3 Four Worker PPO

The four worker PPO assumes the action space with 4 groups of actions. These are manoeuvre actions, self-sensor actions, entity related fire-sensor actions and entity related message action. All worker has independent policy function and critic. The main advantage of this scheme is that it allows multiple tactical discrete actions together. Another advantage is that it makes the joint implementation of imitation learning (supervised learning) and RL much feasible because the multiple actions in the rule-based behaviour should not be allowed for single action imitation learning, which is a necessity to continue RL training by utilizing the policy of imitation learning as a starting point. Within four worker approach, each worker can be considered as experts in their own field. Each worker is rewarded based on their separate reward function and each reward function provides the related motivation to corresponding worker. Thus, workers will be able to focus on the reward signals of the environment that concern them.

5.0 EXPERIMENTS

In this part, experimental setup, experimental details, evaluation metrics and experimental results are

provided.

5.1 Experimental Setup

Some modifications are implemented on the existing FIVE simulator software in order to implement the trainings and make inference from the trained behaviour model. In order to understand the modifications, the design elements created for the experiment, the hardware and software details used within the scope of the project are explained in this section.

Training algorithms are used to smarten up virtual entities in simulations. These algorithms enable virtual entities to learn optimum behaviours. Training algorithms should be designed in order to transform the experiences of the entities in the simulator into artificial intelligence functions (i.e., policy functions) via neural networks which models behaviour rule of the virtual entity. After being trained with training algorithms, rule-based virtual entities turn into learning-based intelligent virtual entities. Three basic elements involved in designing simulators equipped with smart virtual entities are shown in Figure 1. As seen in the figure, the simulator interacts with intelligent virtual entities it contains. In order to transform the experiences of virtual entities into policy functions, it is necessary to determine the state from the simulator and make a decision in the light of this state. The information state coming to the virtual entities and the actions taken by them are input to the training algorithms.

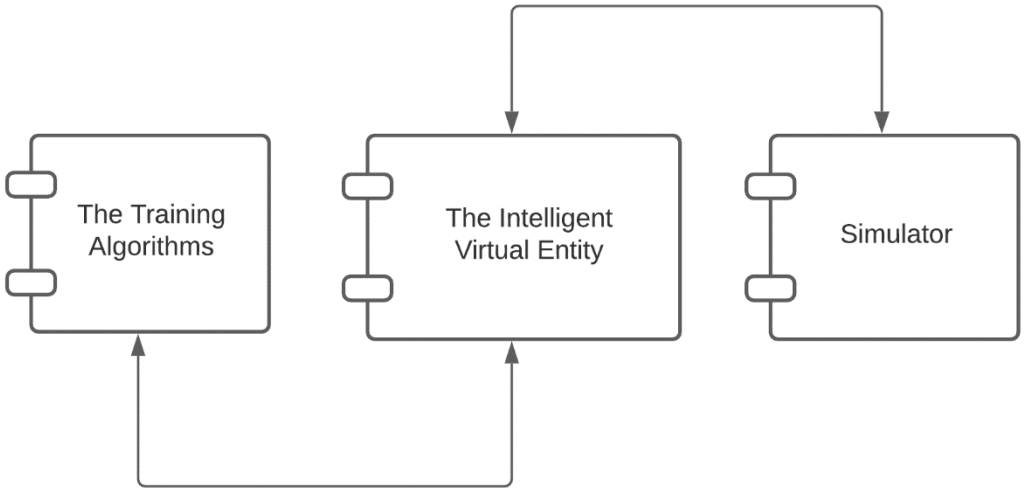

Figure 1 The main elements of the FIVE-ML system design

The structure in simulation design consists of the combination of three sub-sections: training server interface, communication interface and physics engine interface. The training server interface is the part that contains the artificial intelligence engine. The communication interface is used to communicate between the training server and the physics engine interface. The physics engine interface is a software block that contains the components of the simulation where the laws of physics are simulated. These relationships are described in Figure 2

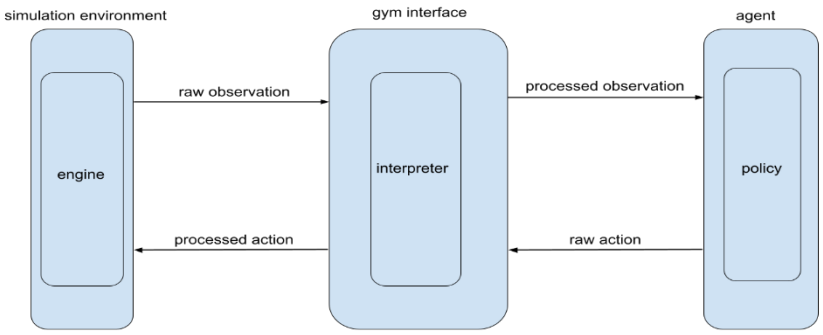

Figure 2 The relationship between the system interfaces

For the trainings, HPE ML350 server is utilized with 2 RTX 3090 GPUs, 128 GB ram and 2 HPE ML350 Gen10 Xeon-S 4215R, each has 8 cores with 3.2 GHz. The current version of FIVE software has an ability to create 8 simulation instances. Therefore, 8 different trainings can be run independently or 8 instances can be utilized for single training. 1K GYM steps are played within approximately half an hour.

5.2 Experimental Details

The trainings are implemented using hybrid PPO and Four-Worker PPO algorithms. For PPO algorithms, episode horizon is limited to 250 steps per agent. Update timestep is set to 4000. Learning rate of policy network is set to 0.0003 and learning rate of the value network is set to 0.001. The hyperbolic tanh function is utilized conveniently with the original PPO baseline [32], [37]. The standard deviation of the continuous PPO model is set to 0.3. The epoch of each update is set to 50. From the multi-agent perspective, decentralized approach is followed where only the local state is available to the agent and actions of each agent are determined separately for each agent (intelligent entity).

For the done condition, other than the episode horizon, two main criteria are determined. One of the criteria is the destruction of any of the intelligent virtual entity which is controlled by RL trained behaviour rule. The other condition is the complete destruction of the enemy entities and there are three enemy entities in the scenarios of this study. For the RL trainings, constant starting location implementation is utilized.

The scenarios in these experiments are air-to-ground and air-to-air scenarios which can be seen in Figure 3 and Figure 4. As seen in the both scenario, there are three enemy and three friend entities. In these scenarios, intelligent behaviour rules are loaded onto the friend F-16 units which are labelled as F-16_01 and F-16_02 in the Figure 3 and Figure 4. The other F-22 entity and enemy entities are controlled by HAVELSAN rule-based behaviour rules. In air-to-air scenario, F-22 aircraft which is commander of the formation detects and determines which hostile entity to be engaged with its long-range advanced radar system. Then commander guides the wingman F-16 aircrafts which are intelligent agents with datalink. The key tactic to destroy the enemy entities is to utilized IR missile within a proper firing range (firing range of IR missile is less than radar missile) instead of radar missile because the rule loaded to enemy entities does not have a defence strategy against IR. In the air-to-ground scenario, the friend entities and the relationship between them is mainly the same. While engaging with ground defence units such as ZSU-23-4, the key to destroy the hostile unit is approaching from behind and leaving the sight area of hostiles so that ground defence system will be ineffective as it cannot deploy turret position properly. The original scenarios are labelled as 3-3 air-to-ground and 3-3 air-to-air scenarios. Additional air-to-ground scenario is also reproduced by removing the F-16_02 entity which is called 3-2 air-to-ground scenario.

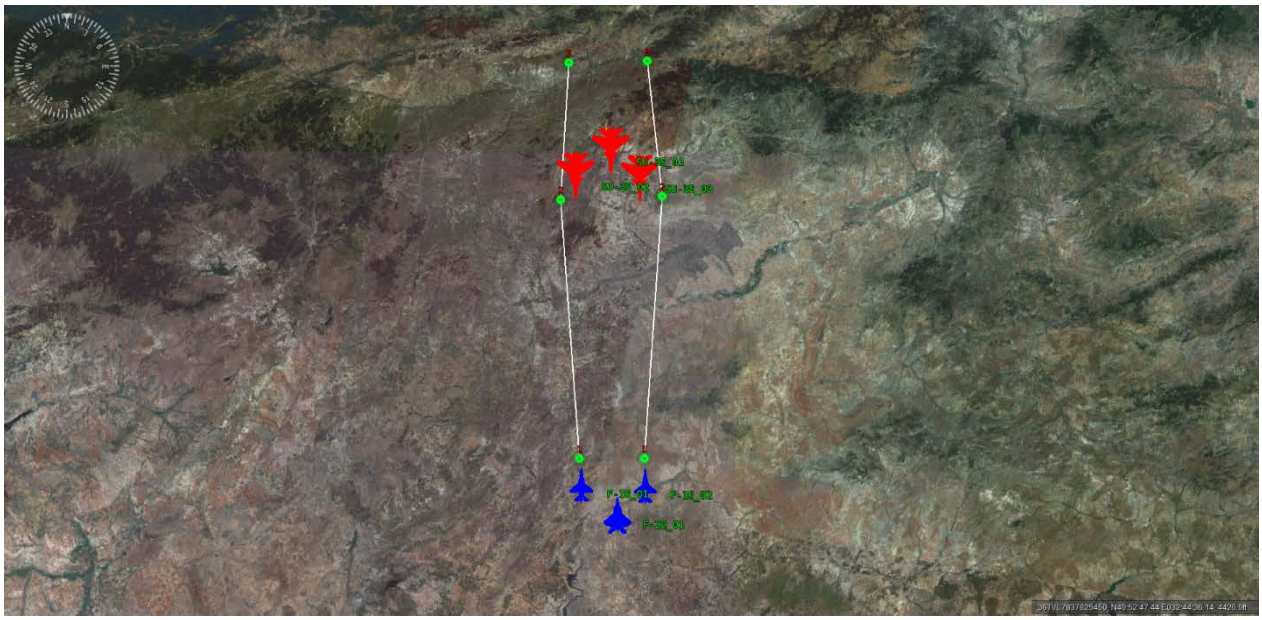

Figure 3 A snapshot from the air-to-air scenario from the tactical environment software

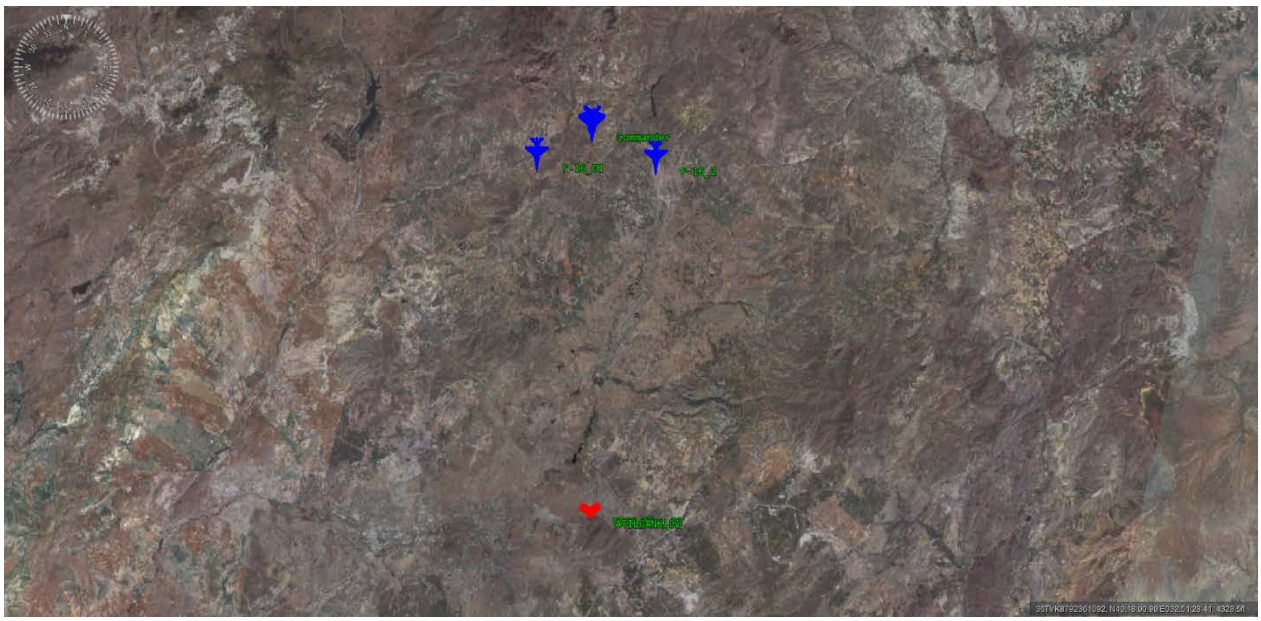

Figure 4 A snapshot from the air-to-ground scenario from the tactical environment software

5.3 Evaluation Metrics

For the evaluation of training performances, two important metrics are determined. These are own hit and enemy hit metrics. Own hit defines or represents the punishment for the hit of agent itself. Enemy hit represents the hit reward of the enemy entities. The impact of other metrics is more negligible than own hit and enemy hit metrics. The main idea of this approach is to show the agent the main goals to achieve throughout the scenario and wait for it to learn from his own experience the tactics agent should apply during the exploration.

The damages of the entities are represented with four damage scales which are from zero to three and three represents the total destruction. For the total destruction of the intelligent virtual entity, the punishment is set to -500 and it is linear with the damage levels. For the total destruction of every enemy entity, the reward is 200 and it increases linearly with the damage levels. The own hit punishment does not change depending on the intelligent virtual entity. However, the maximum of enemy hit reward changes with the number of intelligent virtual entities which is one or two in the experiments of this study. The reason for this change is that every intelligent entity gets a reward for every enemy hit it scores. However, for the last kill, only the one enemy gets the enemy hit reward because of done criteria. Therefore, for 3 enemy and 1 intelligent entity case, the maximum hit reward is 600, while it is 1000 for 2 intelligent entity case. For the logging of metrics, wandb [38] machine learning development tools is utilized.

5.4 Experimental Results

In this section, the experimental analysis of RL trainings is covered. For this part, 3-2 air-to-ground, 3-3 air-to-ground and 3-3 air-to-air scenarios are utilized. The metrics which are made use of for the analysis are enemy hit rewards and own hit punishments. The results in this part are shown in the smoothed format because the original metrics are very noisy due to the exploration techniques existing in RL architectures.

The result of metrics for 3-2 air-to-ground and 3-3 air-to-air scenarios are shown in Figure 5 and Figure 6, respectively. As seen in the figures, the agent or agents can learn to reduce the own hit punishments and increase the enemy hit reward during the training progress. However, it should be denoted that the enemy hit reward increasing in a decreasing rate and algorithm cannot reach the maximum enemy hit reward (which are 600 and 1000 points respectively) and maximum own hit punishment (which is 0). For 3-2 air-to-ground training, enemy hit reaches to 513 points out of 600 after 140K steps. For 3-3 air-to-air training, enemy hit reaches 913 out of 1000 points after 700K steps. As a result of these trainings, the intelligent entities learn to turn on jammer or released chaff as a defence strategy against radar missiles. It learns to choose convenient missile type according to the target entity. It learns to utilize the missiles effectively with respect to firing range. It learns limited manoeuvre capability.

Another analysis that is implemented is the utilization of the trained policy of 3-2 air-to-ground scenario as a starting point 3-3 air-to-air scenarios. This analysis is shown in Figure 7. Firstly, it can be observed that, despite the small change in state space, agent can preserve the learnt abilities to some extent because enemy hits and own hits are started from a good point. The starting point of enemy hit reward of 3-2 air-to-ground trained policy in 3-3 air-to-ground is approximately 800 points out of 1000. After 200K trainings, the agents get about 930 points out of 1000. From the own hit punishment perspective, there is not a significant change because pre-trained policy has already learned to survive.

Another analysis that is covered in this study is the joint implementation of imitation learning and RL. For this analysis, 3-2 air-to-ground scenario is selected. Then, with imitation learning by utilizing supervised learning techniques, HAVELSAN rule-based behaviour rule is imitated via supervised learning. Additionally, imitation learning is implemented on the data gathered from the different locations of the environment, therefore, the policy has a better generalization capability against location change, therefore has a better manoeuvre capability. Then from this starting point, RL algorithms are trained. The result of this analysis is shown in Figure 8. Although the supervised learning model has 400 enemy hit points at the beginning, it reaches 590 enemy hit points out of 600 after 400K steps. This is an important result because this is the highest enemy hit ratio compared to previously mentioned trainings. For the supervised learning part, 4-worker PPO is utilized.

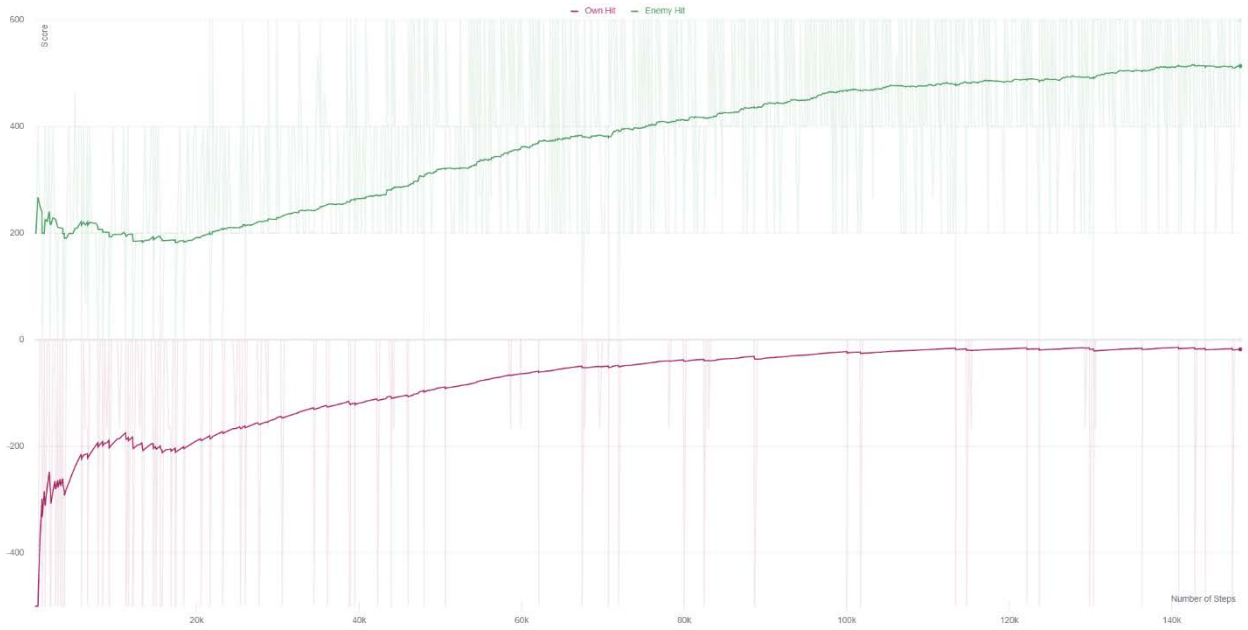

Figure 5 Smoothed score graph of enemy hit rewards and own hit rewards in 3-2 air-to-ground scenario training. On the graph, enemy hits that increase at a decreasing rate are shown in the graph above (plotted in green), while saturating own hit scores are shown in the graph below (plotted in red).

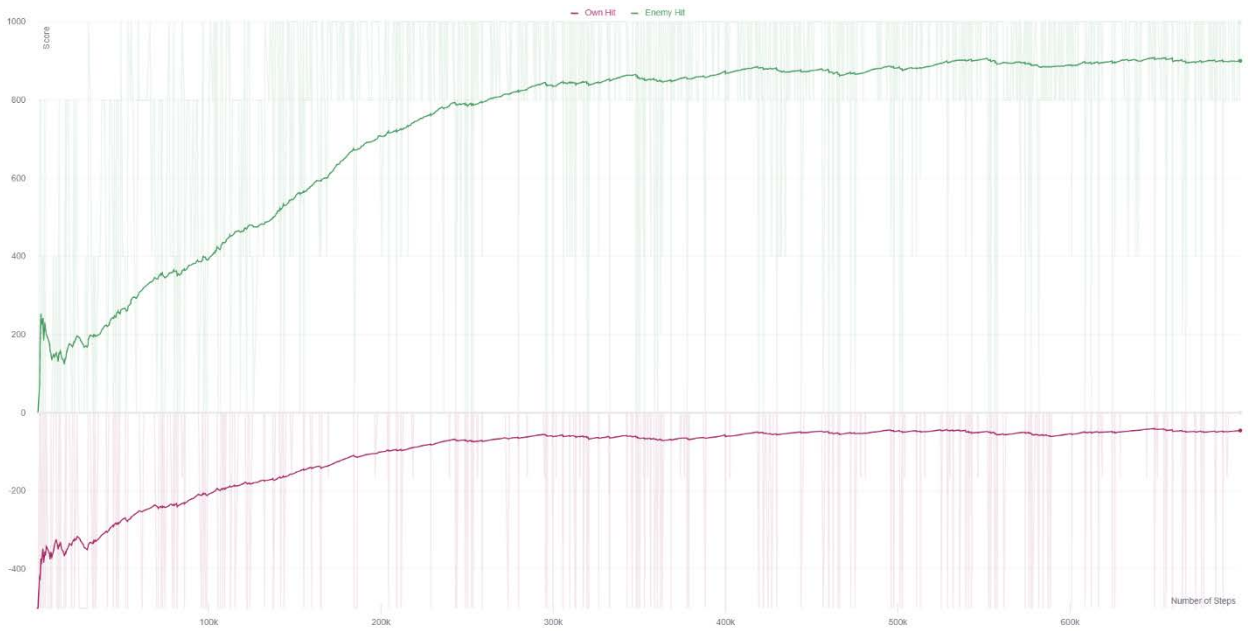

Figure 6 Smoothed score graph of enemy hit rewards and own hit rewards in 3-3 air-to-air scenario training. Enemy hits that increase at a decreasing rate are shown in the graph above (plotted in green), while saturating own hit scores are shown in the graph below (plotted in red).

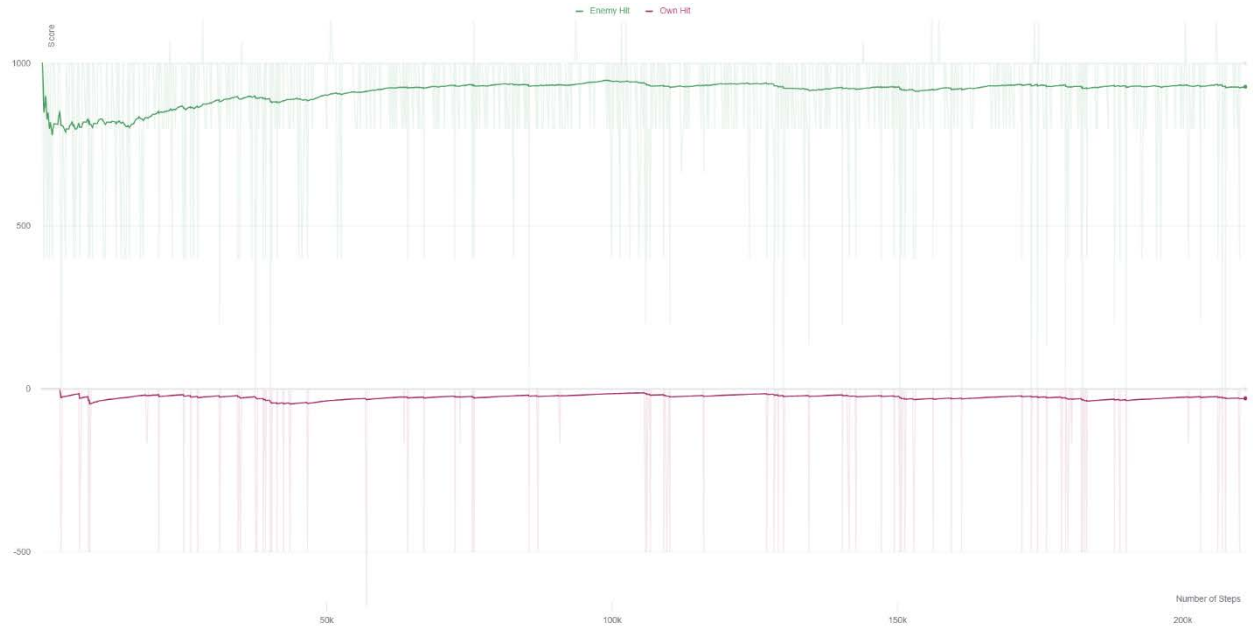

Figure 7 The reinforcement learning training performance of 3-3 air-to-ground scenario from the starting pre-trained policy model of 3-2 air-to-ground scenario with RL training. On the graph, enemy hits are shown in the graph above (plotted in green), while already saturated own hit scores are shown in the graph below (plotted in red).

Figure 8 The reinforcement learning training of 3-2 air-to-ground scenario from the starting pre-trained policy model of 3-2 air-to-ground scenario with supervised learning training. On the graph, enemy hits are shown in the graph above (plotted in green), while already saturated own hit scores are shown in the graph below (plotted in red).

6.0 CONCLUSION

In this work, namely FIVE-ML, the transition from currently existing the rule-based behaviour models of FIVE software to learning-based intelligent behaviour models are analysed. For this aim, firstly the training structure to FIVE software is modified and developed further in order to run RL algorithms on the software. With the result of first-stage experiments, it is shown that RL-based behaviour models can outperform the rule-based HAVELSAN behaviour models in both air-to-air and air-to-ground scenarios. Moreover, it is also shown the concept of imitation learning with RL can also be utilized which is crucial to reduce the training times.

As a future work, the study will be expanded to other scenario types such as air to air defence, air to ground SEAD, and ground to air defence. Moreover, the complexity of the scenarios is planned to be increased by appreciating the number of virtual entities and intelligent virtual entities. Moreover, comprehensive trainings will be implemented in order to provide a better robustness against the changes in the type and class of the enemy entities, the changes in the location, heading and altitude of all entities.

REFERENCES

[1] J. E. Laird, “An exploration into computer games and computer generated forces,” in *Eighth Conference on Computer Generated Forces and Behavior Representation*, 2000.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[3] J. Gauci, E. Conti, and K. Virochsiri, “Horizon: An open-source reinforcement learning platform - Facebook Engineering,” 01-Nov-2018. [Online]. Available: https://engineering.fb.com/2018/11/01/ml-applications/horizon/. [Accessed: 16-Mar-2021].

[4] O. Vinyals *et al.*, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[5] OpenAI *et al.*, “Dota 2 with Large Scale Deep Reinforcement Learning.” 2019.

[6] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, “Self-critical Sequence Training for Image Captioning,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1179–1195, Dec. 2016.

[7] J. Jin, C. Song, H. Li, K. Gai, J. Wang, and W. Zhang, “Real-Time Bidding with Multi-Agent Reinforcement Learning in Display Advertising,” 2018.

[8] Z. Zhou, X. Li, and R. N. Zare, “Optimizing Chemical Reactions with Deep Reinforcement Learning,” 2017.

[9] S. Wang, D. Jia, and X. Weng, “Deep Reinforcement Learning for Autonomous Driving,” *[RecSys2018]Proceedings 12th ACM Conf. Recomm. Syst.*, pp. 95–103, Nov. 2018.

[10] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[12] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” Apr. 2018.

[13] J. Carreira and A. Zisserman, “Quo Vadis, action recognition? A new model and the kinetics dataset,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 4724–4733.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Oct. 2018.

[15] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara, “Meshed-Memory Transformer for Image Captioning,” Dec. 2019.

[16] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Advances in Neural Information Processing Systems*, 2016, pp. 4572–4580.

[17] C. Finn, S. Levine, and P. Abbeel, “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization,” *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 1, pp. 95–107, Mar. 2016.

[18] T. Teng, A. Tan, W. Ong, and K. Lee, “Adaptive CGF for pilots training in air combat simulation,” in *2012 15th International Conference on Information Fusion*, 2012, pp. 2263–2270.

[19] J. Roessingh *et al.*, “Modelling CGFs for tactical air-to-air combat training Motivation-based behaviour and Machine Learning in a common architecture,” 2011.

[20] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. Van Den Herik, “Rewarding Air Combat Behavior in Training Simulations,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 1397–1402.

[21] J. Källström and F. Heintz, “Multi-Agent Multi-Objective Deep Reinforcement Learning for Efficient and Effective Pilot Training,” in *Proceedings of the 10th Aerospace Technology Congress, October 8-9, 2019, Stockholm, Sweden*, 2019, vol. 162, pp. 101–111.

[22] J. Källström and F. Heintz, “Agent Coordination in Air Combat Simulation using Multi-Agent Deep Reinforcement Learning,” in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 2157–2164.

[23] Y. Zuo, K. Deng, Y. Yang, and T. Huang, “Flight Attitude Simulator Control System Design based on Model-free Reinforcement Learning Method,” in *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2019, pp. 355–361.

[24] J. Ted, “Tactical Simulation in Air-To-Air Combat: Evolutionary Algorithms and Behavior Tree Framework,” Luleå University of Technology, 2018.

[25] N. A. Centre, “Smart Bandits,” 1059 CM Amsterdam, 2018.

[26] J. Lindberg, “Simulation driven reinforcement Improving synthetic enemies in flight simulators,” Linköping University, 2020.

[27] W. Kong, D. Zhou, Z. Yang, Y. Zhao, and K. Zhang, “UAV Autonomous Aerial Combat Maneuver Strategy Generation with Observation Error Based on State-Adversarial Deep

Deterministic Policy Gradient and Inverse Reinforcement Learning," *Electronics*, vol. 9, no. 7, p. 1121, Jul. 2020.

[28] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, and J. Liu, "UAV Air Combat Autonomous Maneuver Decision Based on DDPG Algorithm," in *IEEE International Conference on Control and Automation, ICCA*, 2019, vol. 2019-July, pp. 37–42.

[29] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[30] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 4, pp. 2850–2869, Feb. 2016.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *NeurIPS*, pp. 1–9, Jan. 2018.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Jul. 2017.

[33] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 4, pp. 2587–2601, Feb. 2018.

[34] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 3, pp. 1889–1897, Feb. 2015.

[35] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, pp. 6380–6391, Jun. 2017.

[36] L. Engstrom *et al.*, "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO," *arXiv*, May 2020.

[37] P. Dhariwal *et al.*, "OpenAI Baselines," *GitHub repository*. GitHub, 2017.

[38] Wandb, "Developer tools for machine learning." 2021.