2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Evaluating Deep Reinforcement Learning for Computer Generated Forces in Ground Combat Simulation

Babak Toghiani-Rizi, Farzad Kamrani and Linus J. Luotsinen
FOI - Swedish Defence Research Agency
SE-164 90 Stockholm, Sweden
Email: {babtog, farkam, linluo}@foi.se

Linus Gisslén
SEED - Electronic Arts
SE-118 28 Stockholm, Sweden
Email: linus.gisslen@ea.se

***Abstract*—Deep learning techniques are able to process and learn from data (e.g., images, video, audio) without explicit feature extraction. As a result, not only is the manual workload to build such models reduced, but the performance and accuracy of these models can often outperform those in which the preprocessing phase embeds human intuition.**

In the light of these advancements this study aims to examine if current, often manual, practices and techniques for modeling tactical behavior can be improved using deep reinforcement learning (DRL). We compare three state-of-the-art DRL algorithms according to their ability to control computer generated forces in simulated ground combat scenarios. The algorithms are empirically evaluated by comparing learning curves and behavioral performance using four basic maneuverability tasks. Our results show that at least one algorithm solved all tasks without hyperparameter search.

I. Introduction

Military training and decision support applications often rely on simulation to manage and maintain realism in warfare scenarios. Commercial tools supporting such applications often ship with mature features to support military end-users in preparation, execution, visualization and analysis of scenarios and simulation runs. However, there is a capability gap concerning the artificial intelligence (AI) features embedded in these tools. Specifically, intelligent agents or non-player characters, which in military terms are known as computer generated forces (CGFs), that populate the scenarios often lack realistic decision making skills. As a result, military training facilities depend on expensive and often scarce human-role players to ensure credibility and realism in exercises. Although CGFs are used in training, they are often scripted and limited to carry out relatively simple tasks that are triggered by predefined events in the scenario.

Recent advancements in AI and specifically deep learning (DL) [1], [2] has revolutionized the performance of a wide range of traditional AI applications (e.g., face recognition and identification [3], lip-reading [4], autonomous driving [5], image description synthesis [6], Q&A systems [7]). The performance gain observed in some of these applications were not expected to be seen until 10 years from today [8].

Although the modeling of truly intelligent CGFs, that can reason and act at a human level for general purposes, remains an open research question we believe that DL has the potential to significantly improve the state-of-the-art of current CGF development practices and applications. For instance, adopting a DL approach has the following potential advantages compared to the traditional, explicit programming, approach:

- Efficiency: Developing behavior models for CGFs is a time-consuming and expensive endeavor where the knowledge of military domain experts has to be manually interpreted, programmed and integrated into the simulation tool used by the end-users. Implementing CGFs capable of interacting and team-working with other actors (e.g., trainees, other CGFs) may take several years to complete. Using a DL approach the manual process of interpreting and programming the CGF is replaced by a semi-automatic process in which the computer's learning algorithm extracts knowledge from data to identify the behavioral rules of the CGF.
- Realism: Typically, CGFs are developed using a combination of knowledge extracted from subject matter experts, doctrines and rules of engagement. This approach is by definition subjective and will most likely also produce subjective CGFs. In previous works we demonstrated that a data-driven approach [9] towards modeling CGF behaviors can be used to imitate/clone behaviors as they are expressed in data to, ultimately, model more objective CGF behaviors [10].
- Complexity: Developing CGFs manually may limit the creativity and problem solving skills of the CGF. Domain experts and programmers are limited by human intuition and prior experience whereas a computer, using machine learning, is able to explore larger solution spaces to find more optimized, or even superhuman, CGF behavior [11].

The true potential for CGFs in military training and decision support applications is yet to be unlocked. In this work we provide insight into the potential of DL applied in the context of CGFs in ground combat environments. We present results from a preliminary study, based on the results of [12], where state-of-the-art Deep Reinforcement Learning (DRL) algorithms are compared and evaluated given four different learning tasks of increasing complexity.

978-1-5386-1644-4/17/$31.00 ©2017 IEEE

II. Related Works

In [13], Mnih et al. experiment with the approach of training Atari-playing agents [14] using a DRL variant of Q-Learning [15] referred to as Deep Q-Learning (DQL). In DQL, the agent learns which actions to perform using only images and the score from the game as inputs. DQL approximates the Q-function using a deep convolutional neural network [2], [16].

Asynchronous Advantage Actor-Critic (A3C) is a more recent DRL algorithm [17]. A3C asynchronously trains the agent by executing multiple agents in parallel each exploring and learning from different parts of the environment. The A3C algorithm can be applied using standard CPUs whereas the computationally heavy DQL approach, in addition, relies on expensive GPUs or massively distributed architectures to facilitate learning.

In [18], Rijken et al. develop CGFs to model the behavior of Unmanned Aerial Vehicles (UAVs) in air-combat scenarios. Their results indicate that DRL can be applied to learn air combat behavior, but that this area needs more research prior to real-world application.

Unlike the abovementioned works this paper focus on the ground combat domain. CGFs in ground combat exercises must in addition to fundamental skills such as aiming, shooting and moving in the environment also manage to collaborate, interact and coordinate their actions with other CGFs, trainees or role-players.

III. Method

To evaluate DRL in the context of ground combat simulation we have applied three state-of-the-art DRL algorithms on four simulated ground combat tasks as described in Section III-A and Section III-B respectively. The experimental setup (i.e. hyperparameters, reward functions, simulator settings and performance evaluation metrics) is presented in Section III-C.

A. Learning algorithms

1) Deep Q-Learning (DQL): The DQL algorithm is implemented as described in [19], however, we will briefly discuss some of its features, which are required to understand the meaning of the hyperparameters given in Section III-C.

The algorithm is a modification of the standard Q-learning method [15], which is widely used in reinforcement learning [20]. A convolutional neural network (henceforth referred to as Q-network) is used to approximate the optimal action-value function (Q-function), which provides the maximum sum of accumulated rewards at each time-step given a policy. Convolutional neural networks belong to a special class of artificial neural networks with multiple hidden layers, which have shown high performance in visual recognition.

Like other Q-learning methods, the algorithm is model-free, meaning that the CGF learns the task directly using samples from the simulator without having knowledge about the dynamics of the environment and the received rewards.

To ensure a balanced exploration rate of the state space, the algorithm follows a greedy policy with probability $1 - \epsilon$ and selects a random action with probability ϵ, where $0 \leq \epsilon \leq 1$ has a higher value at the beginning of the learning process, when more exploration is required, and decreases successively. During the training, ϵ starts at ϵ_{start} and decreases linearly to ϵ_{end} over a number of simulation ticks, $t_{annealing}$, and is fixed at ϵ_{end} thereafter.

To mitigate instability in the training process, the DQL algorithm relies on two techniques. The first one is using a *replay memory*, which stores previous experiences (i.e. a set of 4-tuples containing state, action, reward and following state) in the memory and reuses them for training by randomly sampling from the memory. The size of this memory, $\mathcal{D}_{size}$, affects both the training time and efficiency of the Q-network, and should be chosen carefully.

The second technique used to further improve the stability of weights of the Q-network is to employ a target network. The target network is a clone of the Q-network, which is used to select actions at each tick. However, it is not updated as frequently as the Q-network but rather after a number of ticks defined by a parameter, C. This adds a delay between the time the Q-network is updated and the time that the update affects the selection of actions, making divergence or oscillations of weights much more unlikely [13].

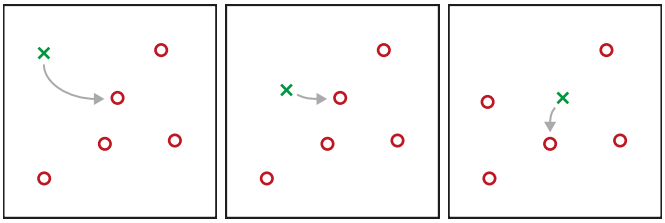
2) Asynchronous Advantage Actor-Critic with a Feed-Forward Network (A3C-FF): Asynchronous deep reinforcement learning algorithms suggested by [17], execute $\mathcal{P}$ number of CGFs in parallel on the same number of separate instances of the environment, training one global network asynchronously. This paradigm removes the need for a replay memory as used in DQL, since each parallel CGF is likely experiencing a different part of the environment at the same time. This has a similar decorrelating effect on the data as randomly sampling from earlier experiences. This parallelism presumably leads to more robust and effective solutions.

Apart from the parallelism nature of the A3C method, we do not attempt to describe it further here, and refer the reader to [17] for a description of the algorithm.

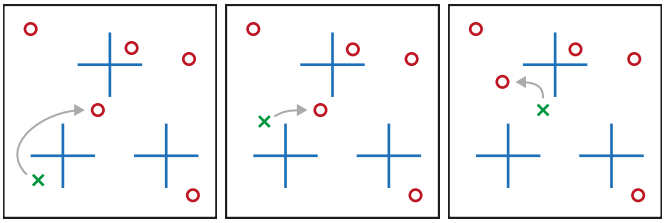
3) Asynchronous Advantage Actor-Critic with an LSTM Network (A3C-LSTM): This method is algorithmically similar to A3C-FF, but differs in the sense that the network includes an additional layer of Long Short-Term Memory (LSTM) cells. LSTM networks are an extension of the recurrent neural network architecture, where the network holds an internal state, which allows it to have temporal behavior. By adding an additional hidden layer of LSTM cells between the fully connected neurons and the output layer, the CGF can potentially learn from a sequence of t_{max} simulation ticks, which could improve its performance in tactical maneuvers that span over a longer period of time.

B. Learning tasks

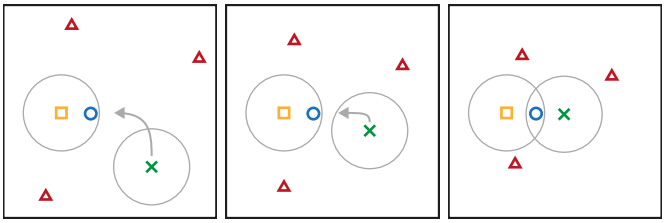
Figure 1 provides an overview of the learning tasks used in this study. In the first task (see Figure 1a), the goal of the CGF is to rendezvous at random positions in an environment. The second task (see Figure 1b) extends the first task by adding obstacles to the environment. In the third task (see Figure 1c), the goal of the CGF is to protect a high value



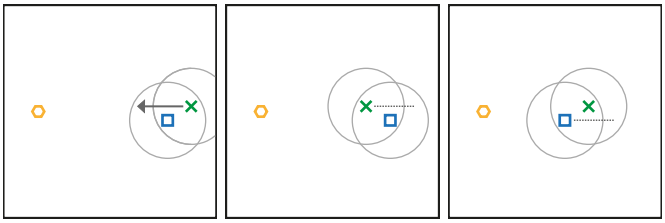
(a) Task 1: Rendezvous (×=CGF, ○=Position).



(b) Task 2: Rendezvous with obstacle avoidance (×=CGF, ○=Position, +=Obstacle).



(c) Task 3: Protect high value individual (HVI) (×=CGF, □=CGF$_{manual}$, ◯=Guard area, ○=HVI, △=Threat).



(d) Task 4: Bounding overwatch movement tactics (×=CGF, □=CGF$_{manual}$, ◯=Guard area, ⬡=Goal).

Fig. 1: Learning tasks used to evaluate DRL in the context of CGF behaviors for ground combat simulation.

individual (HVI) from threats. In this task, the CGF must learn to collaborate or adapt its behavior, given the behavior of another, manually implemented, CGF. Finally, in the fourth learning task (see Figure 1d), the goal is to learn the bounding overwatch movement tactic [21]. In bounding overwatch two entities collaboratively move towards a shared destination by alternating their movements. When the first entity is moving the other is overwatching and vice versa. This tactic is used to move safely in high-risk areas where enemy presence is likely.

In this work an in-house developed simulator is used to evaluate the DRL algorithms. The simulator provides an interface from which the DRL algorithms can inject CGF actions, retrieve a reward signal and, finally, extract image sequences representing the CGF's perception of the environment. The simulator implements a basic action set that allows the CGFs to move in four directions within the simulated environment (a_{east}, a_{west}, a_{north} and a_{south}). The action set also includes an overwatch action, $a_{overwatch}$, which is used in the bounding overwatch learning task as illustrated in Figure 1d. The simulator also embeds explicitly programmed CGFs representing own, enemy and neutral forces as required by the learning tasks.

C. Experimental setup

1) Neural network: The convolutional neural network used by the three DRL algorithms in this paper is configured as illustrated in Figure 2. The input layer takes as input 4 images from the 4 latest simulation ticks. Each image consists of 80×80 pixels. That is, the input layer consists of $4 \times 80 \times 80 = 25600$ neurons each representing a gray scale pixel value. The first hidden layer consists of 16 feature maps generated by 8×8 kernels. The second hidden layer consist of 32 feature maps generated by 4×4 kernels. The third hidden layer is fully connected using 256 neurons. Finally, the output layer consists of one neuron per available action, so 4 or 5 neurons depending on the learning task. The neural network structure in Figure 2 illustrates the architecture used in DQL and A3C-FF. The A3C-LSTM network has an additional hidden layer of 256 LSTM cells between the 256 fully connected neurons and the output layer.

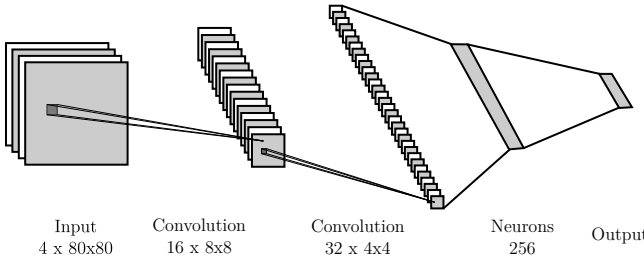


Fig. 2: Network structure used in the DQL and A3C-FF algorithms.

2) Hyperparameters: The hyperparameters for the DRL algorithms discussed in Section III-A are listed in Table I. The training of each CGF was set to a maximum number of simulation ticks as specified in [19] (DQL) and [17] (A3C), but was also constrained to maximum running time of 24h.

3) Reward functions: The reward function in the first task rewards the CGF when reaching within $d_{min} = 10$ pixels of any O_i positions in the environment. If the CGF attempts to move outside the simulated environment, it is negatively rewarded. The second task shares the same reward function as the first task, with the exception that $d_{min} = 5$, to make the task more difficult. The reward function for the reward, r, in learning tasks 1 and 2 can be viewed in Equation 1.

DQL		
Variable	**Hyperparameter**	**Value**
C	Target network update freq.	10 000
ϵ_{start}	Initial exploration rate	1.0
ϵ_{end}	Final exploration rate	0.1
$t_{annealing}$	Annealing period	1 000 000
$\mathcal{D}_{size}$	Replay memory size	1 000 000
A3C-FF		
Variable	**Hyperparameter**	**Value**
$\mathcal{P}$	Parallel CGFs	16
A3C-LSTM		
Variable	**Hyperparameter**	**Value**
$\mathcal{P}$	Parallel CGFs	16
t_{max}	LSTM sequence length	5

TABLE I: A subset of the hyperparameters for the DQL and A3C algorithms. All other hyperparameters were set according to [19] and [17] respectively.

$$r = \begin{cases} 1 & d(O_i, CGF) < d_{min} \\ -1 & \text{moving outside the environment} \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

In the third learning task, the CGF is rewarded when it eliminates threats. However, eliminating threats while at the same time leaving the HVI unguarded returns only a small reward, since the HVI will be at risk for being approached by the remaining threats. Thus, the task of the CGF is to learn to collaborate with the manually implemented CGF, CGF_{manual}. If the HVI is in the intersection of the guarding areas of the CGF and CGF_{manual}, both CGFs will be able to eliminate threats, ultimately returning a higher reward calculated based on how much area the CGFs covered collaboratively. Since the guarding area of each CGF has a radius of 10 pixels, their Euclidean distance need to be less than $d_{max} = 20$ to protect the HVI. The reward function for the reward, r, in learning task 3 can be viewed in Equation 2.

$$r = \begin{cases} r_d & d(threat_i, CGF) < 10 \wedge HVI \text{ is guarded} \\ 0.1 & d(threat_i, CGF) < 10 \wedge HVI \text{ is unguarded} \\ -1 & d(threat_i, HVI) < 5 \\ -1 & \text{moving outside the environment} \\ 0 & \text{otherwise,} \end{cases}$$

where

$$r_d = \frac{d(CGF_{manual}, CGF)}{d_{max}} \quad (2)$$

In the fourth learning task, the CGF is rewarded based on successful overwatch triggers and reaching the goal. The CGF can trigger the overwatch only if it is within the other CGF's guarded area, resulting in a reward based on the squared Euclidean distance of the advanced distance, d_{adv}, of CGF_{manual}. Limited by the radius guarding area, this distance can at max reach $d_{max} = 20$. The reward function for the reward, r, in learning task 4 can be viewed in Equation 3.

$$r = \begin{cases} r_d & \text{if } action = a_{overwatch} \wedge CGF_{manual} \text{ is guarded} \\ 1 & \text{if } d(goal, CGF_{manual}) < 10 \\ 0 & \text{otherwise,} \end{cases}$$

where

$$r_d = \left(\frac{d_{adv}}{d_{max}}\right)^2 \quad (3)$$

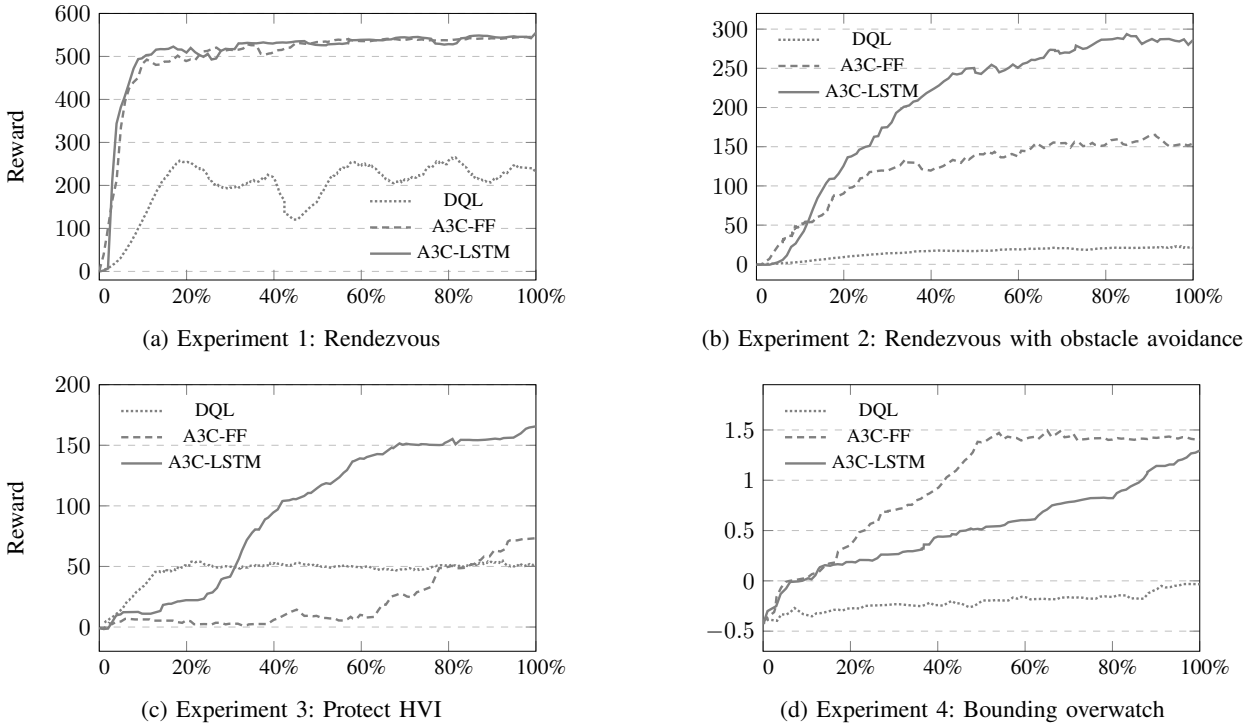
4) Performance evaluation metric: To evaluate the performance of the trained CGFs, the reward signal from 1000 randomly initiated simulation runs is measured for each learning task. For each simulation run the maximum number of simulation ticks was set to 2000.

IV. Results

In this section we provide results from our experiments. The line-plots in Figure 3 illustrate how the average reward changes over time, t, for each algorithm and task during training. In this plot, $t = 0\%$ and $t = 100\%$ represents start and end of training respectively. The line-plots provide insight into the algorithms' ability to learn each task. The box-plots in Figure 4 represent the reward statistic (median, min, max, lower quartile and upper quartile) after training, i.e. $t = 100\%$, from 1000 randomly initiated simulation runs. The box-plots provide insight into how well the algorithms are able to generalize. High variance, represented by large boxes in the plot, generally indicates poor generalization capabilities whereas low variance and small boxes represent good generalization capabilities.

Experiment 1: In the first experiment, where the algorithms are applied to learn the rendezvous task, there is a distinct learning performance difference between the DQL algorithm and the two A3C algorithms. In Figure 3a we observe that A3C-FF and A3C-LSTM averaged a reward of 500 at $t = 10\%$ and then improved to approximately 550 at the end of training. The DQL algorithm averaged a reward value of approximately 250 at $t = 20\%$ and maintained this value, with some variation due to exploration annealing, to the end of training. The box-plots in Figure 4a indicate that the A3C variants are able to generalize whereas DQL struggles to adapt to new, randomly initiated, environments.

Experiment 2: In the second experiment the algorithms are applied to learn how to rendezvous while at the same time avoiding obstacles. In this case the A3C-LSTM algorithm outperformed A3C-FF as illustrated in Figure 3b. A3C-LSTM averaged a reward of approximately 300 at the end of training whereas A3C-FF only managed to average a reward value of 150. The DQL algorithm, although slowly increasing its reward throughout training, failed to learn this task. DQL only averaged a reward of 20 at the end of training. From the box-plots in Figure 4b we observe that A3C-LSTM is able to generalize well whereas A3C-FF struggles to perform in randomly initiated environments.



(a) Experiment 1: Rendezvous

(b) Experiment 2: Rendezvous with obstacle avoidance

(c) Experiment 3: Protect HVI

(d) Experiment 4: Bounding overwatch

Fig. 3: Line-plots representing the average reward over time, where $t = 0\%$ and $t = 100\%$ represents the start and end of training respectively, for each learning algorithm and experiment.

Experiment 3: In the third experiment the learning task is to collaboratively protect an HVI from threats. From Figure 3c we can observe that the averaged reward of A3C-FF and DQL at the end of training are relatively low compared to A3C-LSTM. From Figure 4c we also observe that A3C-LSTM is the only approach that is able to generalize.

In this experiment positive rewards are given when threats are eliminated and negative rewards are given when threats manages to move too close to the HVI. A3C-LSTM learned to, at all times, stay close to the HVI whereas DQL and A3C-FF at times moved away, leaving the HVI vulnerable to threats.

Experiment 4: In the fourth experiment the task is to learn bounding overwatch. In this experiment DQL failed, see Figure 3d, to learn the bounding overwatch movement tactic. DQL ended up with an averaged reward near 0 after training. The learning curves of A3C-FF and A3C-LSTM diverge slightly since A3C-FF converged faster. The box-plots also indicate that they perform similarly with respect to generalization capabilities.

V. Discussion

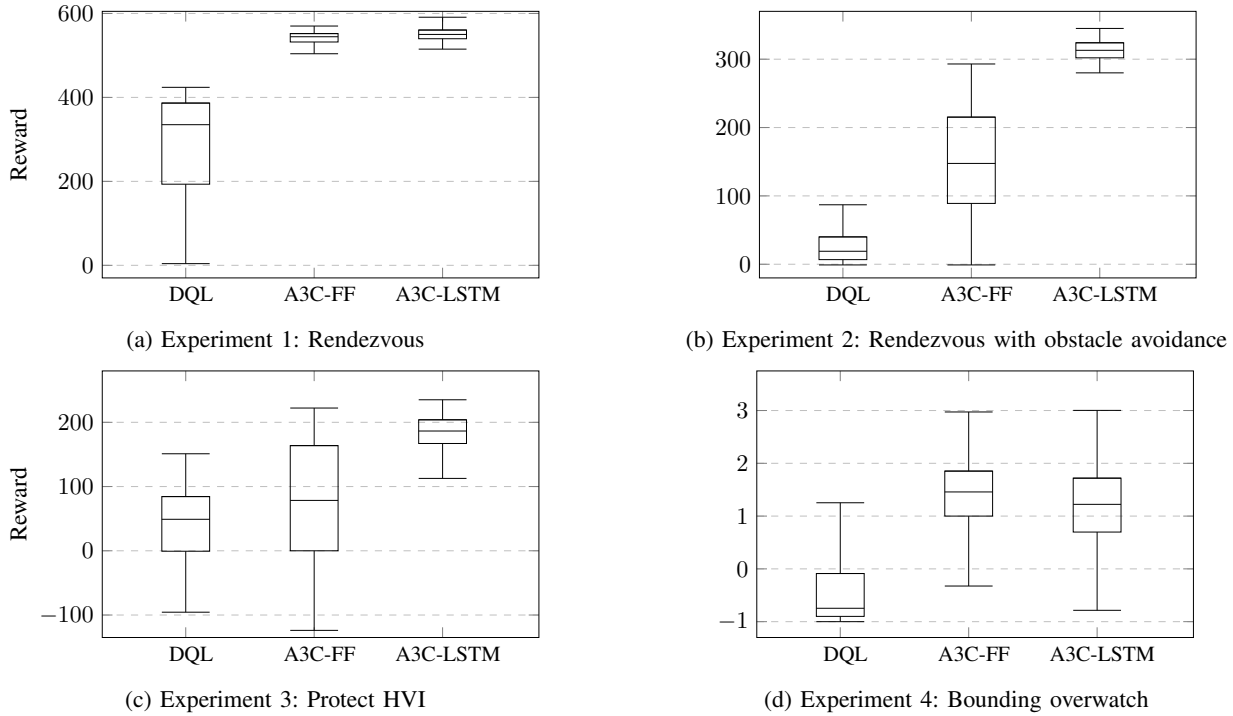
Experiments show that A3C-LSTM successfully learned all tasks within reasonable training time. The temporal ability (enabled by the LSTM part of the algorithm) increased the overall reward for learning tasks 2-3 over A3C-FF. The DQL approach could not match the performance of A3C-LSTM in any of the tasks. We hypothesize that the DQL algorithm needs more training time to converge to valid solutions.

Here the algorithms learn the simulated tasks directly from raw images and feedback (i.e. reward) provided by the simulator. Even though these tasks are scriptable by a human we argue that the algorithms can potentially learn more complex tasks. The advantage with self-learning algorithms is therefore two-fold: the ability to efficiently come up with solutions to a given task without involving human intervention, and in an online setting where new situations arrive which cannot be pre-scripted.

VI. Conclusions

In this work we have evaluated three DRL algorithms and their ability to learn basic maneuverability tasks in simulated ground combat scenarios. For each task we designed and implemented a simulator capable of providing: image data representing the state of the environment; an action repertoire that allows for the CGF to move in the environment; and, a reward function to signal good or bad decision making. Our evaluation shows that A3C-LSTM was able to successfully learn all tasks. DQL did worst and was unable to learn any of the tasks to the level of A3C-LSTM. A3C-FF performed well but was unable to match A3C-LSTM in tasks 2-3.

We conclude that DRL techniques has the potential to improve current CGF behavior modeling practices. In future works we intend to evaluate A3C-LSTM using a real-world ground combat simulator.



(a) Experiment 1: Rendezvous

(b) Experiment 2: Rendezvous with obstacle avoidance

(c) Experiment 3: Protect HVI

(d) Experiment 4: Bounding overwatch

Fig. 4: Box-plots representing reward statistics from 1000 simulation runs. The plots were generated using the model found after training, $t = 100\%$, for each learning algorithm and experiment.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1701–1708.

[4] J. S. Chung and A. Zisserman, "Lip reading in the wild," in *Asian Conference on Computer Vision*, 2016.

[5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 2722–2730.

[6] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3128–3137.

[7] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: an overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, 2010.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[9] L. J. Luotsinen and R. A. Løvlid, "Data-driven behavior modeling for computer generated forces," in *NATO modelling and simulation group symp. M&S support to operational tasks including war gaming, logistics, cyber defence (MSG-133)*, 2015, pp. 1–13.

[10] F. Kamrani, L. J. Luotsinen, and R. A. Løvlid, "Learning objective agent behavior using a data-driven modeling approach," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 2175–2181.

[11] L. J. Luotsinen, F. Kamrani, P. Hammar, M. Jändel, and R. A. Løvlid, "Evolved creative intelligence for computer generated forces," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 3063–3070.

[12] B. Toghiani-Rizi, "Evaluation of deep learning methods for creating synthetic actors," Master's thesis, Uppsala University, 2017.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, 2012.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI*, 2011, pp. 1237–1242.

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[18] R. Rijken and A. Toubman, "The future of autonomous air combat behavior," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 3089–3094.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *NIPS Deep Learning workshop*, 2013.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[21] Unites States Department of the Army, "The infantry rifle platoon and squad, field manual no. 3-21.8," 2007.