УДК 629.7.05

RULKO E. V.

ACTOR COMPLEXIFICATION IN TD3 AND CURRICULUM LEARNING WITH STRUCTURAL COMPOSITION FOR DRONE COUNTERING

Military Academy of the Republic of Belarus
Minsk, Republic of Belarus

Abstract. *The work suggests complexifying actors within the framework of TD3 which involves the usage of different state vectors for actors and critics in order to assure convergence of the algorithm. It also describes a process of aggregating models, separately trained on datasets or in simulation on tasks with increasing difficulty, stitching everything together step by step into a single end-to-end system. It allows utilizing existing algorithms, such as YOLO, in reinforcement learning systems, performing sensor fusion and gradually adding functionality without losing convergence. Assistance providing allows training systems in simulation from hardcoded algorithms that use simplified states. These techniques are demonstrated on a particular task of building an anti-drone system for armored vehicles.*

Keywords: *deep reinforcement learning, TD3, curriculum learning, sensor fusion, UAV*

Introduction

Training of modern robotic systems requires ample multifarious data that can only be obtained in simulation. There are different ways of creating simulated environments. NVIDIA Omniverse [1] is a platform that was specially built for such purposes. Isaac Sim [2] is an application built on that platform that enables developers to simulated and test AI-driven robotic solutions in physically based virtual environments – Figure 1.

Figure 1. Robots in simulation [3]

There are also game development engines such as Unity3d [4] and Unreal [5] that can be used for such purposes. Plethora of high-quality photorealistic 3D models of military objects allows training advanced computer vision systems for military purposes – Figure 2.

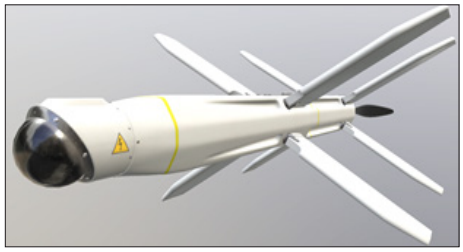

Figure 2. 3D model of a drone [6]

Simulation enables to train systems of arbitrary configurations with different weather, terrain, lighting and noise conditions.

The main part

In the current work we use Unity3d and explore the process of gradual composition and training for an anti-drone turret equipped with LIDAR, two microphones and two cameras. Turret is mounted on a vehicle and has a cannon to shoot drones – Figure 3. Principles, described in the work, can be applicable for systems of more complex configurations, different sensors and ways to counter drones. Precise cannon control and decision making about firing are performed based on video feed. Audio and LIDAR data is necessary for crude initial orientation towards a target and early warning.

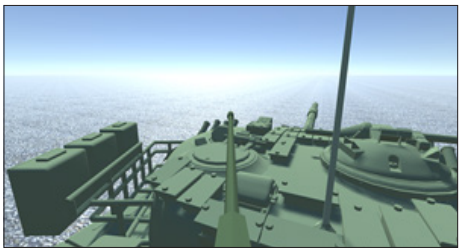

Figure 3. View from a turret camera

For ensuring convergence of the entire system, we start from a simple case with a stationary vehicle and a single target. The first trainable part is responsible for processing LIDAR data. Point cloud is converted to a depth map which is used as a state vector for reinforcement learning (RL) algorithm that guides a cannon towards a single target in simulation. Such gimmick is connected with necessity of writing your own LIDAR in Unity3d ML-Agents [7] and we use a grid sensor [8] that simulates that instead – Figure 4.

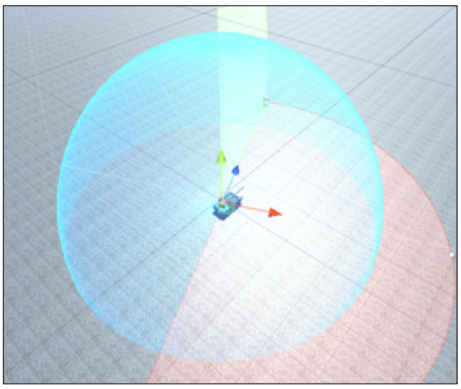

Figure 4: Imitation of LIDAR coverage

Current environment requires continuous actions and a loss in this case is based on the angles discrepancies. Different RL algorithms were considered as a possible foundation for the system and tested with a simplified task to guide a canon towards a randomly moving target, given just a set of previous angles and a current angle towards it. The list included Soft Actor-Critic (SAC) [9], Proximal Policy Optimization (PPO) [10], Deep Deterministic Policy Gradient (DDGP) [11] and Twin Delayed Deep Deterministic Policy Gradient (TD3) [12]. In the result, TD3 turned out to fit the task well because of its deterministic nature that is apt for cannon guidance and the ability to regulate exploration noise on different training stages. Delayed policy update, target policy smoothing and clipped double Q-learning provide more stability in learning in comparison with its predecessor DDPG. Networks of different architectures, with different sets of hyperparameters were tested for actors and critics. During this stage, we have two continuous actions as output (altitude and azimuth shifting) – Figure 5 (activation and maxpool layers are discarded).

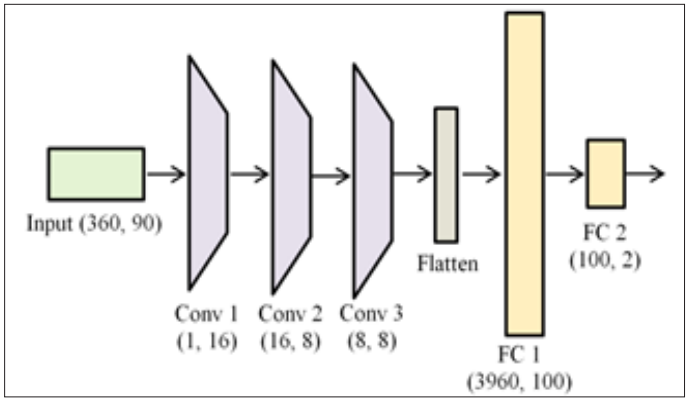

Figure 5. Actor, based on a convolutional network

In spite of the fact of quick converging with just angles as an input state, processing a depth map of a relatively small resolution 360x90 and controlling turret based on that, turned out to be an unsurmountable task for TD3 with not pre-trained networks or either required extensive period of time on a single machine, considering the necessity to simultaneously run a simulation as a computational bottleneck.

During the research, different ways to make it converging were tested, including Assistance Providing [13] from a TD3 initially trained only on angles and a usage of two-headed networks for actors and critics, additionally taking angles with increasing dropout over training in order to be "weaned off" from "cheating" – Figure 6.

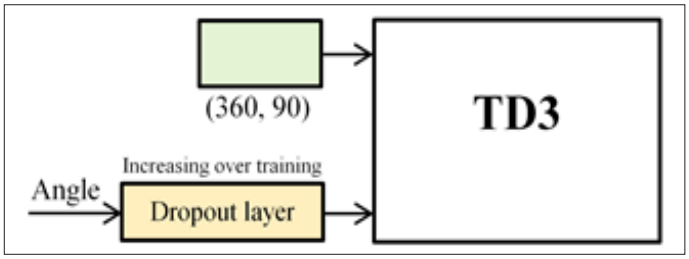

Figure 6: "Cheating" TD3 with increasing dropout

In that case the networks initially relied on the angle value and when a dropout was close to 1 the training process collapsed. Eventually a working solution was found. It involves training actors and critics with simpler state vectors S_1 (angles) and then using those pretrained and frozen critics together with actors that are provided with more complex states S_2 (depth map) – Figure 7.

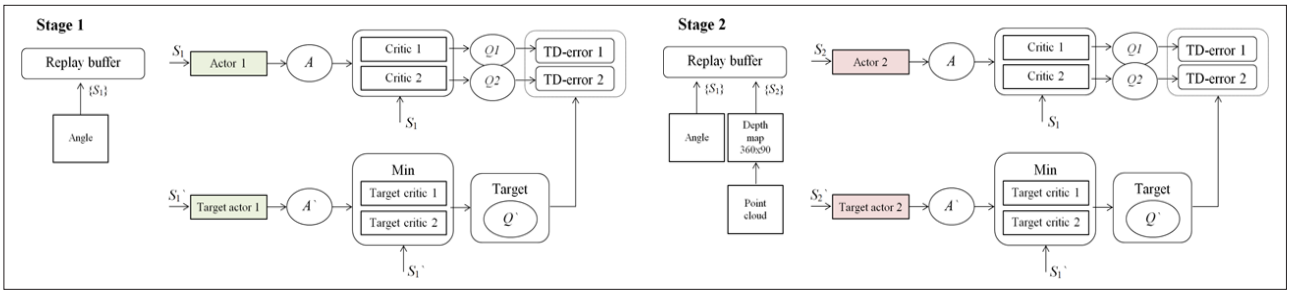

Figure 7: Actor complexification

The approach of actor complexification can be not limited to a single iteration. After the second state, we can freeze the actors and train the critics on the depth map, and then complexify the state of actors again – Figure 8.

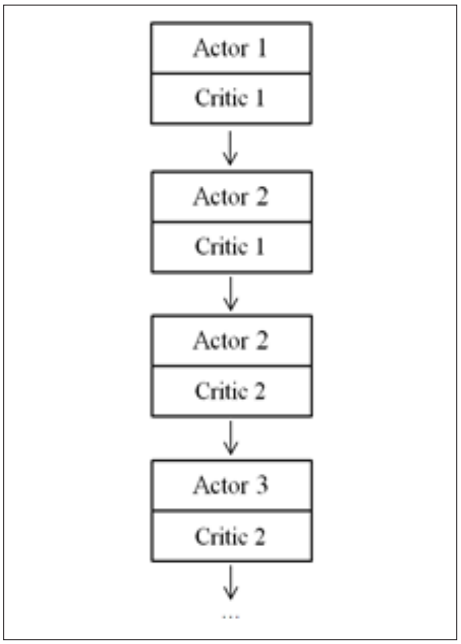

Figure 8: Way of complexification

Training a RL system in simulation that utilizes sound requires constant processing of Mel spectrograms, that hinders training. A more savvy way is to pretrain networks in a supervised mode on existing datasets, and then incorporate them into the RL algorithm as heads of more complex networks (actors and critics) and perform only sensor fusion and fine tuning in simulation. That approach is applicable for range of modalities. There are open datasets containing environmental sounds [14] as well as datasets with sounds of drones of different types [15, 16]. Resultant dataset for a binary classification task (drone, not drone) was combined by overlaying sounds of drones with a plethora of sounds, including sounds of military vehicles, and also overlaying together sounds without drones. Two primary architectures were considered: based on pretrained ResNet50 and pretrained visual transformer vit_b_16 [17] available with PyTorch – Figure 9.

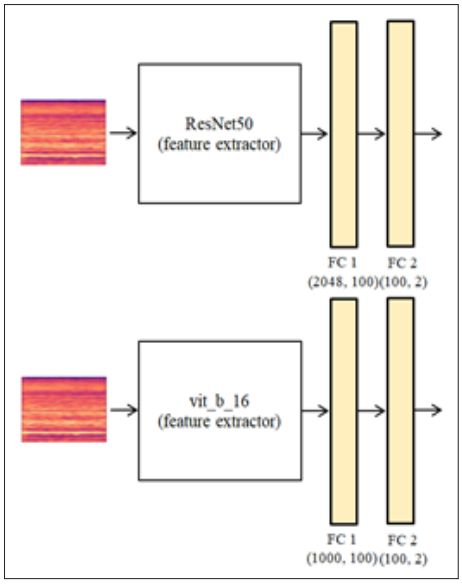

Figure 9. Drone detection networks

It turned out that the networks can confuse some environmental sounds like mosquitos, bees or ordinary planes with drones, which required further work with the dataset – Figure 10.

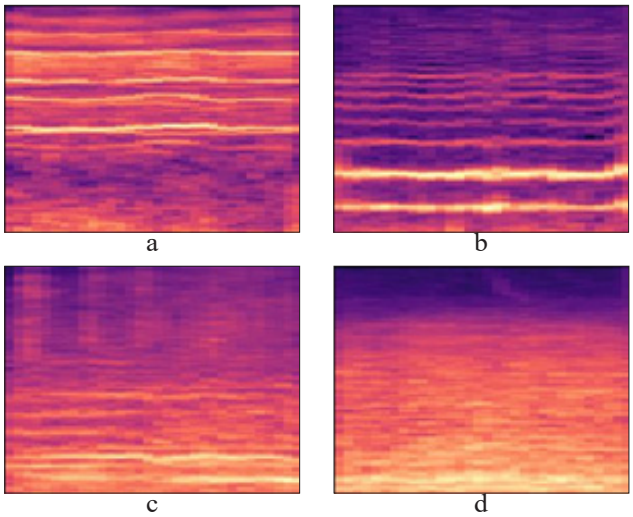

Figure 10. Mel spectrograms of a drone (a), mosquito (b), bee (c) and a passenger jet (d)

There are extensive datasets, containing varieties of insects sounds like [18, 19] as well as with sounds of planes [20]. It transpired that just labeling mosquito sounds as not drone led to a situation when a simultaneously flying mosquito and drone would be identified as not drone. That required mixture of all possible drone sounds with sounds of different insects with the results labeled as drone and mixture of different insect sounds together labeled as not drone. The entire dataset was almost 1 million Mel spectrograms. The visual transformer based network demonstrated 99.8 % accuracy with an average time required to process a second of sound of 0.012 sec on NVIDIA GeForce RTX 2080 SUPER, and ResNet50 based network demonstrated 97.9 % accuracy with a time of 0.011 sec (only processing by a network).

The next stage is to orient a turret towards a drone by pure sound. A dataset, combined of environmental sounds with drone sounds added to each channel, is used for that. The two headed network for left and right stereo channels, based on visual transformer, trained during the previous stage (Figure 9), predicts a value between -1…1 depending on the volume of a drone sound in each channel – Figure 11.

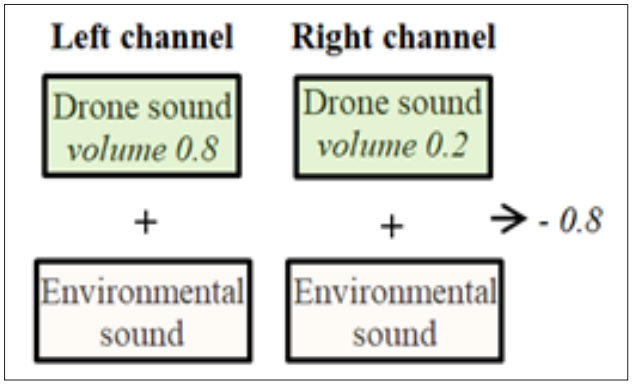

Figure 11. Dataset creation

The next stage is performing sensor fusion of LIDAR and sound heads in simulation for a system that will direct the turret – Figure 12. Actor complexification (Figure 7) is used further to assure convergence.

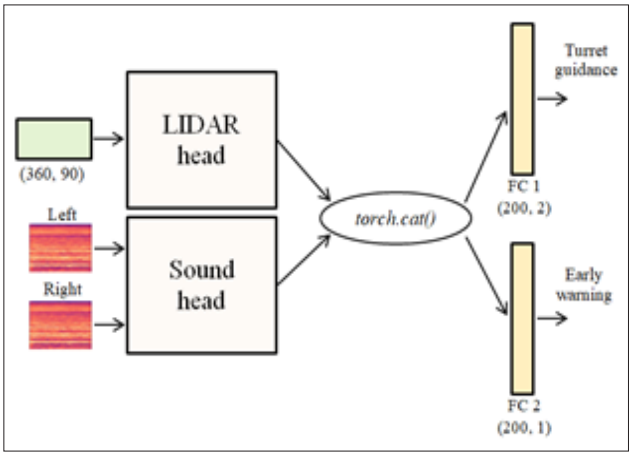

Figure 12. Combined heads

This combination of heads is done for both actors and critics; in this project they have the same architecture, but critics also take actions as input and they are responsible for Q-function instead of a policy. Heads are taken from previously described networks, discarding last fully connected layers. That's why for reasons of equitable contribution in terms of feature richness each of networks finishes with a layer that takes 100 input features. Each weight of those layers is divided by 2 and loaded into FC1 [13]. "Early warning head" is responsible for activating a system in order not to rotate a turret without drones in the vicinity. Training is conducted in alternated cycles. During the training of the early warning head, the guiding head is frozen and a drone flies with different trajectories, velocities and sounds or may not fly at all in order to make a network determining a fact of presence of a drone. During the training of the guiding head, the early warning one is frozen and a drone flies all the time in order to guide a turret into its direction. On this stage the sound is grabbed from simulation, and unlike the LIDAR data which is constantly updated, Mel spectrograms are built only each second.

Slicing Aided Hyper Inference (SAHI) with YOLOv11 is used for processing frames from both zooming and wide angled cameras, which is trained initially in a supervised mode to detect only one class (drone) on datasets with drones like [21], birds [22] and synthetic data obtained using Unity Perception [23] from 3D models of drones surrounded by different environmental objects.

In a hypothetical case of a single attacking drone it would be reasonable to provide a network with a center of detected object, but the need to repel an attack of several drones makes it necessary to determine the order of engaging and facts of defeating each target. That necessitates estimating distances by analyzing dynamics of angular size changing (LIDAR may not cover long enough distance) and converting set of bounding boxes to an image containing just those boxes with the brightness proportional to their confidence and track IDs for making it easy to cope with multiple targets. Feature vector from wide angle camera is added to LIDAR and sound features, and during the current stage the system learns to track a single target with frozen YOLO weights.

The next stage requires involvement of a zooming camera. In this case a zooming head is rewarded based on keeping the full bounding box of a drone as big as possible within its view and punished if a bounding box goes beyond view even partially, with everything else frozen.

Curriculum step for firing head involves rewarding it for hitting the target, and lightly punishing for the usage of ammo, and strongly punishing for firing outside the projection of the target. The fact of hitting the target is determined by simulating bullet dispersion from the aiming point. In order to make the system adaptable to different cannon configurations and types of ammo, the firing head takes as input parameters of measured ballistic dispersion and number of rounds in a single burst. It can also take different parameters, like the fragment density for grapeshot rounds. That allows the head to "make decisions"

about the best moment to open fire.

When the system is focused on a target, it's necessary to assess its damage after firing, disengage and switch to another target, when the previous one is shot down. Damage assessing head contains LSTM layers and takes the information from both camera heads in order to analyze changing in trajectory and appearance of the engaged target and is trained in a separate curriculum with simulations of drone destructions.

The next stage involves training everything together to cope with single targets. Reward is no longer based on the angle difference, but on eliminating targets, not allowing them to reach the object, taking into account the proximity at which the target was destroyed and the ammunition consumption. The entire structure of the system is presented on Figure 13.

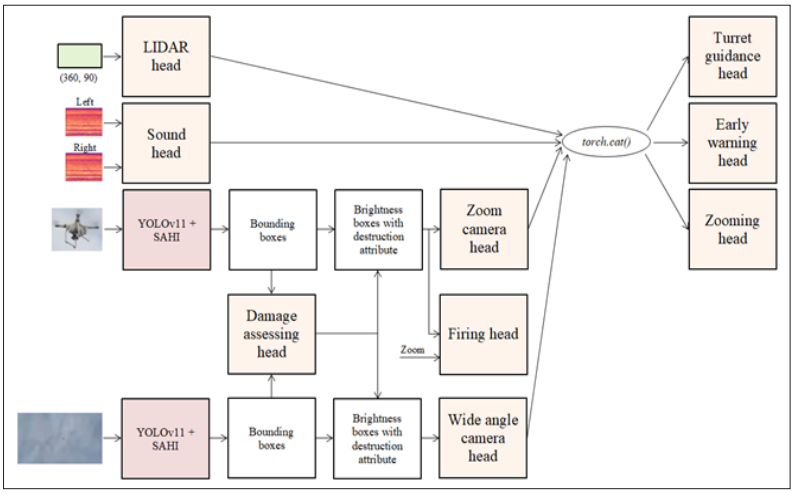

Figure 13. Structure of the system

To ensure convergence, a hardcoded algorithm that uses a proportional–integral–derivative (PID) controller to guide a cannon towards a target, zooms in and fires when a target is in the crosshairs, is used with alternating assistance providing [13], switching between the hardcoded algorithm and the trained system over the training – Figure 14.

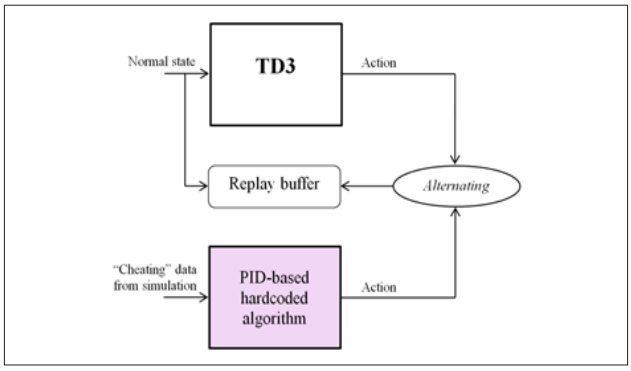

Figure 14. Training from hardcoded algorithm

Such kind of a gimmick allows convergence on complex tasks.

The next stage is to teach the system to counter multiple targets. Earlier the system was able to guide a turret towards a single target based initially on sound and LIDAR data, then capture it by a wide angle camera, zoom on it and make a decision whether fire or not, while tracking it. Now the introduction of several simultaneous targets leads to activation of the system by the early warning head and erratic rotation of the turret. The system must check targets in precedence of their importance based on distance or dynamic of their approaching. And whether engage, if it's a real drone, or move to the next one in case it's a bird or something else. That represents a complex RL task like [24] and necessitates the next phase of extensive training. Modified version of the previous hardcoded algorithm (Figure 14), that engages targets consequently, is used to expedite convergence through assistance providing [13].

Necessity to estimate speed with which targets approach also entailed considering six consecutive frames of LIDAR data, performing feature extraction utilizing weights of a previously trained head, followed by a transformer network – Figure 15.

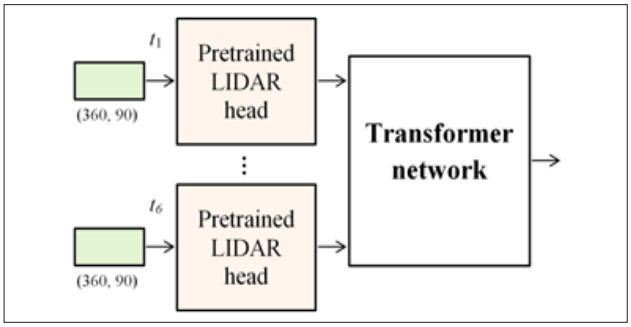

Figure 15. Reconfiguring LIDAR head

That in turn required repeating curriculum steps for training a LIDAR head again with everything frozen

and reward based on angles discrepancy, but eventually it improved performance of the whole system in terms of ability to engage first targets that are plummeting right now towards the vehicle, ignoring remote or stationary ones.

On this stage of curriculum every occlusion will be interpreted by the LIDAR head as a possible target. The system must understand that moving around objects that touch the surface don't represent a threat, and when the vehicle moves, passing above tree branches are also not moving targets. In order to achieve that, each head is modified to accept vehicle's velocity and orientation (with existing pretrained weights [13]), and the entire curriculum cycle is repeated again, including training the early warning head. The environment this time includes naturally possible objects like other vehicles, people, buildings, trees, bushes and so on, as well as different types of drones and birds, each with a distinct sound signature.

Utilizing a trained actor with the real hardware that has mass, inertia, stiffness, sensor inaccuracy and noise can be challenging. For that such hardware parameters must be set in simulation. For example Unity3d provides ArticulationBody component [25] that allows setting parameters such as linear damping, angular damping, joint friction, inertia, center of mass and other. The network can also be trained to adapt to specific hardware by getting a feedback from actuator sensors and environment. This is achieved by initially training the network to manage an actuator (turret) with fixed parameters (inertia, friction, etc.) getting observations that additionally include exact parameters of the actuator (angle, velocity) and then using actuators with slightly different parameters, making the network learn to compensate it – Figure 16. This is analogous to getting accustomed to driving a new car, its steering, suspension, maneuverability, acceleration and brakes. Passing the turn and overtaking are essentially the same using a cope or a minivan, but effort and magnitude of movement of a steering wheel and a gas pedal are different. In our case the network must "understand" that the turret has certain inertia and manage that accordingly.

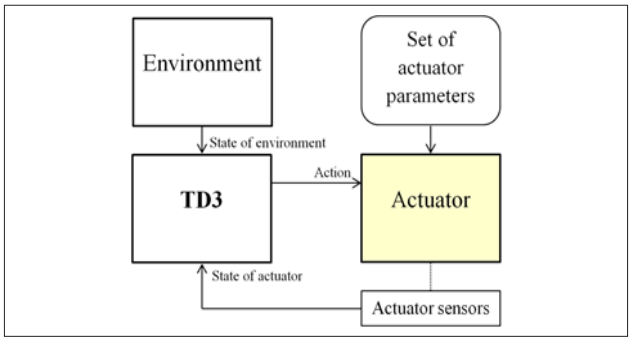

Figure 16. Adaptation to hardware

Additions and further development

It's fair to say that drone warfare is developing with an unprecedented rate and it may be possible and expedient to utilize drones with rocket launchers, that don't get too close to the target, making them difficult to shoot.

During the process of training networks in simulation, drones were flying towards a target straight from where they were spawned and then some random maneuvers were added to make it more difficult to shoot them, at the same time insuring that drones eventually move towards the target. A better way is to train an adversarial system, where a swarm of drones learns how to cooperate and attack shrewdly, figuring out about some tricks of team work to circumvent the enemy. That may be achieved through wielding of Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [26], but providing each critic (Q-network) with the combined perspective of all agents in a team makes it computationally hard to train – Figure 17.

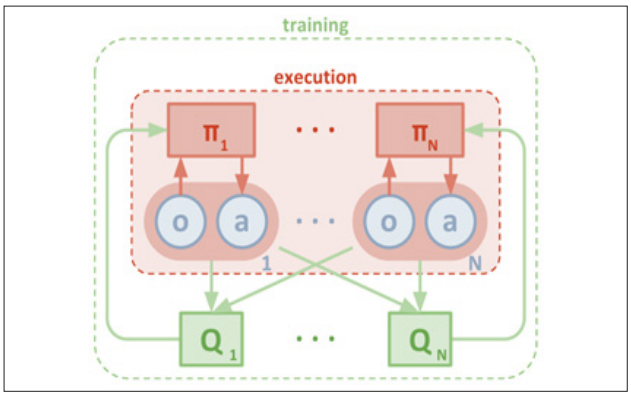

Figure 17. MADDPG [26]

Firing cannon at drones also may not be the best solution to counter them. A vehicle may carry a set of interceptor drones for hitting other drones or dogfighting with weapons, which can also be trained in simulation.

It also seems reasonable to use a pixel-to-voxel projection [27], capturing frames from a set of cameras on stationary objects, or determining cameras' locations and orientations "on the fly", given some recognized key objects that are visible simultaneously by multiple cameras on vehicles, that have communication channels between each other – Figure 18.

Recent achievements in development of multimodal large language models and agentic systems makes it possible to provide AI systems with high level reasoning that is important in some cases of drone countering.

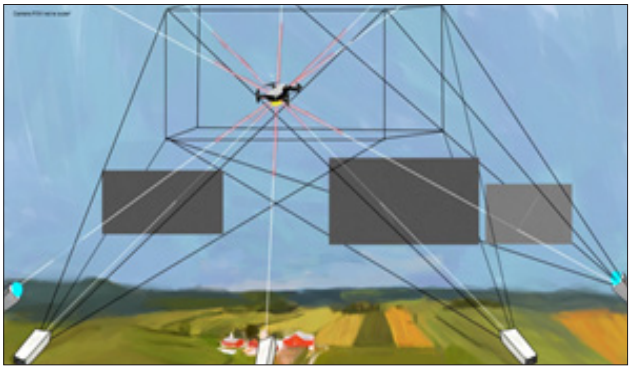

Figure 18. Pixel-to-voxel projection [28]

Another direction of development is to provide AI systems with the ability to build models of the environment and provide sensible ways of interpreting observations based on such models utilizing Free Energy Principle [29].

Conclusions

The suggested approach of actor complexification gives a way to converge RL models utilizing actors and critics with different state vectors, which can be used to interactively increase the level of complexity.

Described process of curriculum learning with structural composition involves aggregating models, separately trained whether on datasets or in simulation on tasks with increasing difficulty within the framework of RL or in a supervised mode.

Usage of hardcoded algorithms that take as input some "cheating" state from simulation allows training complex RL systems through assistance providing [13].

Adaptation to specific mechanical actuators can be performed by training a network that accepts some feedback from sensors and adjusts its actions.

Building an adversarial system would involve training a swarm of drones in simulation that learn to cooperate and overcome defense, which in turn will precipitate the training process for countering unexpected tricks.

Nowadays there is a plethora of existing solutions for designing advanced military AI systems.

REFERENCES

1. NVIDIA Omniverse. Available at: https://www.nvidia.com/en-us/omniverse (accessed: 08 August 2025).

2. NVIDIA Isaac Sim. Available at: https://developer.nvidia.com/isaac/sim (accessed: 08 August 2025).

3. Narrowing the Sim2Real Gap with NVIDIA Isaac Sim. Nvidia. Available at: https://www.youtube.com/watch?v=VW-dOMBFj7o (accessed: 08 August 2025).

4. Unity. Available at: https://unity.com (accessed: 08 August 2025).

5. Unreal engine. Available at: https://www.unrealengine.com/en-US (accessed: 08 August 2025).

6. Loitering Munition Lancet Drone 3D Model. Renderhub. Available at: https://www.renderhub.com/sergeydzyuba/loitering-munition-lancet-drone (accessed: 08 August 2025).

7. Unity ML-Agents Toolkit. GitHub. Available at: https://github.com/Unity-Technologies/ml-agents (accessed: 08 August 2025).

8. Grid Sensors for Unity ML-Agents. GitHub. Available at: https://github.com/mbaske/grid-sensor (accessed: 08 August 2025).

9. Haarnoja T., Zhou A., Abbeel P., Levine S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018. arXiv: 1801.01290. https://doi.org/10.48550/arXiv.1801.01290

10. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal Policy Optimization Algorithms. 2017. arXiv: 1707.06347. https://doi.org/10.48550/arXiv.1707.06347

11. Lillicrap T.P., Hunt J.J, Pritzel A., Heess N., Erez T., Tassa Y., et al. Continuous control with deep reinforcement learning. 2019. arXiv: 1509.02971. https://doi.org/10.48550/arXiv.1509.02971

12. Fujimoto S., Herke van Hoof, Meger D. Addressing Function Approximation Error in Actor-Critic Methods. 2018. arXiv: 1802.09477. https://doi.org/10.48550/arXiv.1802.09477

13. Rulko E.V. Complexification through gradual involvement and reward providing in deep reinforcement learning. System analysis and applied information science. 2024;4:13-20. https://doi.org/10.21122/2309-4923-2024-4-13-20

14. ESC-50: Dataset for Environmental Sound Classification. GitHub. Available at: https://github.com/karolpiczak/ESC-50/tree/master (accessed: 08 August 2025).

15. DroneAudioDataset. GitHub. Available at: https://github.com/saraalemadi/DroneAudioDataset (accessed: 08 August 2025).

16. Drone Audio Detection Samples. Hugging Face. Available at: https://huggingface.co/datasets/geronimobasso/drone-audio-detection-samples (accessed: 08 August 2025).

17. Vit_b_16. PyTorch documentation. Available at: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.vit_b_16.html (accessed: 08 August 2025).

18. Smart Bee Colony Monitor: Clips of Beehive Sounds. Kaggle. Available at: https://www.kaggle.com/datasets/annajyang/beehive-sounds (accessed: 08 August 2025).

19. Time series classification. Dataset: MosquitoSound. Time series classification website. Available at: https://www.timeseriesclassification.com/description.php?Dataset=MosquitoSound (accessed: 08 August 2025).

20. Audio Dataset of Low-Flying Aircraft: AeroSonicDB. Kaggle. Available at: https://www.kaggle.com/datasets/gray8ed/audio-dataset-of-low-flying-aircraft-aerosonicdb (accessed: 08 August 2025).

21. YOLO Drone Detection Dataset. Kaggle. Available at: https://www.kaggle.com/datasets/muki2003/yolo-drone-detection-dataset (accessed: 08 August 2025).

22. Caltech-UCSD Birds-200-2011. Kaggle. Available at: https://www.kaggle.com/datasets/veeralakrishna/200-bird-species-with-11788-images/data (accessed: 08 August 2025).

23. Perception Package. Unity documentation. Available at: https://docs.unity3d.com/Packages/com.unity.

perception@1.0/manual/index.html (accessed: 08 August 2025).

24. Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, et al. Dota 2 with Large Scale Deep Reinforcement Learning. 2019. arXiv: 1912.06680. https://doi.org/10.48550/arXiv.1912.06680

25. Articulation Body component reference. Unity documentation. Available at: https://docs.unity3d.com/6000.1/Documentation/Manual/class-ArticulationBody.html (accessed: 08 August 2025).

26. Lowe R., Wu Y., Tamar A., Harb J., Abbeel P., Mordatch I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. 2020. arXiv: 1706.02275. https://doi.org/10.48550/arXiv.1706.02275

27. Projects motion of pixels to a voxel. GitHub. Available at: https://github.com/ConsistentlyInconsistentYT/Pixeltovoxelprojector (accessed: 08 August 2025).

28. Tracking faint objects with cheap cameras. Consistently inconsistent. Available at: https://www.youtube.com/watch?v=m-b51C82-UE&t=98s (accessed: 08 August 2025).

29. Friston K. The free-energy principle: a unified brain theory? Nature Reviews Neuroscience. 2010;11:127–138. https://doi.org/10.1038/nrn2787

РУЛЬКО Е. В.

УСЛОЖНЕНИЕ АКТОРОВ В TD3 И ОБУЧЕНИЕ ПО КУРРИКУЛУМОМУ СО СТРУКТУРНОЙ КОМПОЗИЦИЕЙ НА ПРИМЕРЕ ЗАДАЧИ ОТРАЖЕНИЯ АТАК БЕСПИЛОТНЫХ ЛЕТАТЕЛЬНЫХ АППАРАТОВ

Военная академия Республики Беларусь
Минск, Республика Беларусь

Аннотация. *В работе предложены усложняющиеся акторы в рамках алгоритма двойного отсроченного глубокого детерминированного градиента политики (TD3), что предполагает использование различных векторов состояний для акторов и критиков с целью обеспечения сходимости алгоритма. Работа также описывает процесс агрегирования моделей, раздельно натренированных на датасетах или в симуляции на задачах с увеличивающейся сложностью, соединяя их вместе шаг за шагом в единую систему. Это позволяет использовать существующие алгоритмы, такие как YOLO, в системах обучения с подкреплением, осуществляя процесс объединения данных датчиков и постепенно увеличивая функциональность без потери сходимости. Предоставление ассистирования позволяет тренировать в симуляции системы машинного обучения на основе жестко запрограммированных алгоритмов, использующих упрощенные вектора состояний. Данные техники продемонстрированы на задаче построения системы защиты бронемашин от БПЛА.*

Ключевые слова: *глубокое обучение с подкреплением, TD3, обучение по куррикулумому, объединение данных датчиков, БПЛА*

Рулько Евгений Викторович
Военная академия Республики Беларусь. Минск, Республика Беларусь.
Кандидат технических наук, доцент. Начальник научно-исследовательской лаборатории моделирования военных действий учреждения образования «Военная академия Республики Беларусь». Сфера научных интересов: глубокое обучение, машинное зрение, обучение с подкреплением, нейронауки, активный вывод, принцип свободной энергии, рефлексивное управление.
Eugene V. Rulko
Military academy of the Republic of Belarus. Minsk, Republic of Belarus.
PhD of Engineering Sciences. Associate Professor. The head of the research laboratory of military operation simulation of the educational institution «Military academy of the Republic of Belarus». Research interests: deep learning, computer vision, reinforcement learning, neuroscience, active inference, free energy principle, reflexive control.
E-mail: eugeni1533@gmail.com