CS231N Project Final Report: Deep Q-Learning applied to tactical deconfliction

Marta Palomar Toledano
mpalomar@stanford.edu

Abstract

This Project consists of the development of a methodology for airborne Conflict Detection and Resolution using Deep Learning. Specifically, we cover the situation of an aircraft (ownship) that has to modify its trajectory to avoid collision with one or more intruders. To this purpose, Reinforcement Learning methods inspired by game theory (Atari Pong) have been implemented in conjunction with a Convolutional Neural Network (CNN) model that takes advantage of input images in order to represent states. This feature is meant to outperform other conflict resolution algorithms as the model is prepared to handle multiple aircraft at the same time in a same state space.

The proposed methodology has been implemented using the OpenAI Gym for custom environment creation and has been tested on different scenarios including potential conflicts with a variable number of aircraft. The rate of success of this tool to resolve separation conflicts has been assessed, as well as the time efficiency of the adopted avoidance maneuvers. Finally, the performance improvement due to the inclusion of a CNN within the DQN algorithm will be evaluated and discussed.

1. Introduction

The scope of this Project is to handle one of the main problems in flying vehicles ecosystems, which is the loss of separation between aircraft. More specifically, the aim is to propose and develop a methodology to conduct tactical deconfliction: when two or more flying vehicles approach each other more than the recommended safety margin, one or both aircraft will perform the most suitable maneuver to avoid collision.

The proposed system's working principle is the following: given a sequence of input images that contain the aircraft in potential conflict, the tool will output the most adequate collision avoidance maneuver. The tool consists of a Convolutional Neural Network (CNN) framework combined with a Deep Q-Learning (DQN) algorithm. Aircraft trajectory data is gathered in the form of images to ease the inclusion of multiple intruders (conflicting aircrafts) within the observation space. The complexity of the problem is large due to various constraints limiting the versatility of trajectory replanning, including aircraft physical limitations, adjusted time schedules, aerodrome availability, obstacle avoidance, etc. Also, the availability of real aerial data from separation conflicts between aircraft is scarce, fact that makes it difficult to obtain a "ground truth" of proposed ATC collision avoidance maneuvers to train and evaluate our methodology.

Such a system will be vital in the future for the insertion of multiple unmanned aerial systems (UAS) into a same airspace in the context of Advanced Air Mobility (AAM) in which, due to the expected high volume of operations, a large number of potential conflict situations may arise and current Air Traffic Management (ATM) capabilities could be insufficient to handle all of them. In fact, the creation of robust Conflict Detection and Resolution (CDR) systems is a requirement set by the ICAO in order to let autonomous vehicles fly beyond visual line of sight [6].

Also, even though in this Project the main focus will be set on handling avoidance of conflicts between aircraft, this same tool could be employed to avoid static obstacles like terrain hazards, etc.

2. Related Work

The problem of aircraft conflict detection and resolution during flight has been addressed in the industry and literature using very different approaches. In the past, the trend was to employ purely-geometrical approaches based on the knowledge of the distance with respect to the intruder (and trajectory data). If such models are based on the trajectories of a fixed number of aircraft, they will not be able to resolve conflicts that imply a variable number of involved vehicles because the input size of the model is fixed. Other approaches taken in the literature include, specially, Optimal control theory [5] and Particle Swarm Optimization (PSO) algorithms [14].

In this Project, in order to overcome some of the existing limitations and allow for a variable number of vehicles involved in a certain scenario, the strategy of converting trajectory information into images has been adopted. This approach has been proposed in prior works [10, 17]. For instance, in [10], their approach consisted of a supervised model based on CNNs (called ACRnet) for air conflict resolution. Their model took as inputs several conflicting aircraft trajectories that were converted to image format and proved to be more accurate than other existing models (CRMLnet) that took trajectory data as inputs without conversion to images. This work highlighted the benefits of using image representation combined with CNNs. Also, the authors of [11] used a convolutional neural network architecture (CNN) which utilized Solution Space Diagram (SSD) images to learn the avoidance maneuvers. The SSDs were artificially created images containing all the variables involved in the conflict. One of the limitations of this work is that the results provide only the initial maneuver and not the complete conflict resolution profile.

One of the novelties included in the current approach is the extrapolation of a DQN architecture currently employed in Atari Pong games [7] to our tactical deconfliction problem. Other published studies have proposed DQN algorithms for collision avoidance for surface or underwater vehicles [3, 15], but their implementation did not rely on the employment of images as inputs. Similarly, a recent study [13] presented an Independent DQN (IDQN) framework to solve the conflict detection and resolution problem in a multi-agent manner. Other Multi-Agent Reinforcement Learning approaches can be found in the literature [2] applied to ATC conflict handling in UAV environments.

Also, some studies handled the CDR topic taking into account the three dimensions of the problem and also time.

3. Methodology

In this Project the purpose to be achieved is the full development of a Deep Q-learning Network architecture capable of solving a tactical deconfliction problem. The implementation will be conducted in Python and it comprises several steps that will be defined in this Section.

3.1. DQN Algorithm

The Deep Q-network (DQN) is one of the most famous methods of deep reinforcement learning [12]. In fact, it outperforms regular Q-learning methods when the problem space increases as DQN gets rid of the two- dimensionality of the Q-Matrix introducing a Neural Network [9]. Even though DQNs can be implemented using only Fully Connected layers, in our case the DQN approximates the action value function by Convolutional Neural Networks (CNNs)

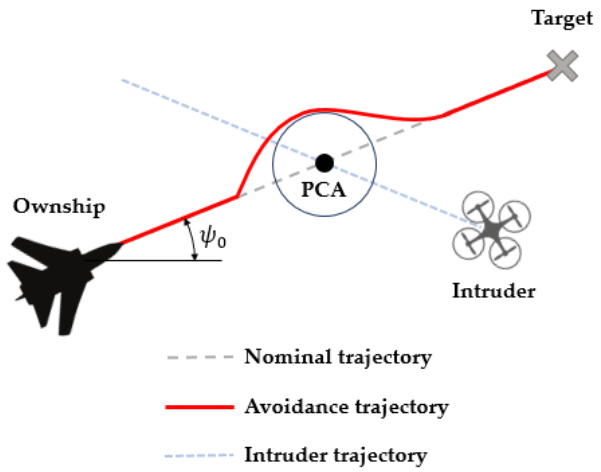

Figure 1. Example of avoidance maneuver and return to nominal heading.

and updates the action value function by Q-learning using the equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \mathrm{argmaxQ(s_{t+1}, a_t)} - Q(s_t, a_t)]$$

where γ is the discount factor and r_t is the reward at each time step. During learning, the model loss function will be dependent on the estimated Q-value function that results from the CNN, using the Q-Learning update relation:

$$L(\theta) = (\underbrace{(r + \gamma \mathrm{max_{a,t+1}Q(s_{t+1}, a_{t+1}; \theta^t))}}_{\text{target Q value}} - \underbrace{Q(s, a; \theta^p)}_{\text{predicted Q value}})^2$$

And the resulting weight update is performed based on:

$$\theta^{t+1} = \theta^t + \alpha[r_{t+1} + \gamma \mathrm{max_{a,t+1}Q(s_{t+1}, a_{t+1}; \theta^t)} - Q(s, a; \theta^p)]\nabla Q(s_{t+1}, a_t; \theta^t)$$

3.2. Python Implementation

In order to ease the structure of the developed code, the OpenAI Gym toolkit [1] for Reinforcement Learning will be employed, as it contains built-in functions that can be easily adaptable to the specific problem to be solved.

3.2.1 Creation of an OpenAI environment

OpenAI Gym provides built-in environments for a variety of RL problems, such as the Atari Pong, and it also allows the

user to create a custom one. In this case, a new Gym environment in which the specific dynamics of the problem are stated, as well as the rendering interface. On the one hand, the initialization conditions are defined and all the possible actions. Also, the *.step* function is created, specifying the movement constraints of the elements and the reward distribution policies, among others.

Elements and dynamics in the environment

The baseline environment created in this project consists of an observation space that is translated to a canvas of size 600x800 px. Every element to be included in the problem is displayed in this canvas, which conforms the state at each timestep, and is passed to an Agent class.

The main elements included in the canvas are two: the ownship, which is the aircraft on which we want to apply the conflict resolution maneuvers, and its target, which is a point representing the ownship's final destination. In this situation, as there are no conflicts, the response of the agent would be just commanding the ownship to maintain its original trajectory until the target is reached. The other element type that can be included is the intruder, i.e. aircrafts whose trajectory is in conflict with the one for the ownship. The number of intruders is adjustable, as well as their time of action during the episode. All the moving elements are defined by three factors: their initial position, their heading, and their speed. For this report only the 2D case will be considered.

We want to ensure that, when including new intruders in the environment space, most or many of them will imply a potential conflict with the ownship nominal trajectory in order to achieve a situation similar to the one illustrated in Figure 1. To this purpose, even though the intruders' initial positions are established randomly at the outer area of the canvas, their heading is set in such a way that their nominal trajectory intersects the ownship's at a certain probability p.

Action Space

According to ATC regulations, controllers' experience and dynamic airspace environment, controllers usually adopt three kinds of resolution methods: altitude adjustments, speed adjustments and heading adjustments [13]. Even though altitude adjustments are some of the most frequent and effective, these are not applicable to the current 2D approach of this Project but should be included in future developments.

For our baseline model, only heading adjustments will be considered although the inclusion of actions modifying the ownship velocity is totally feasible. The action space considered for this Report gathers four options:

- action 0: keep current heading value.
- action 1: increase heading by 30º.
- action 2: decrease heading by 30º.
- action 3: adopt the heading value towards the target.

For the Milestone report of this Project, only actions 0, 1 and 2 were considered and, even though the ownship was able to avoid conflict just with this actions, in many of the cases it was difficult to get to the target once the conflict was resolved. Also, in the cases where the ownship reached the target, the resulting trajectory was suboptimal from a flight time perspective. For this reason action 3 has been included, resulting in a significant increase in the number of episodes that terminate successfully with the ownship reaching its target and at earlier times.

Rewards

Reward functions drastically change AI behavior in simulations and often in unexpected ways. Care must be taken in designing such functions to accurately capture risk and intent [16]. The reward function implemented to our agent has been changed throughout the Project development. At the Project Milestone, the reward function was only taking into account the separation between the ownship and the intruders, and even though conflicts were resolved by taking corrective actions, the resulting trajectories were intricate and time-inefficient.

In order to achieve more efficient trajectories, a point target that the ownship has to reach has been included along the nominal initial trajectory. If the ownship goes out of the observation space without having reached the target, a negative reward is obtained. Also, when the number of timesteps needed to reach the target is larger than an optimum value, a penalty is applied to the reward function.

The reward function for conflict resolution is the same than the applied for the Project Milestone:

$$r(a) = \begin{cases} 0 & d_{intruder} > d_{critical} \\ -\dfrac{d_{critical} - d_{intruder}}{d_{critical}} & d_{intruder} \leq d_{critical} \end{cases}$$

which sets a zero-reward when the distance between the ownship and a certain intruder ($d_intruder$) is larger than an arbitrarily chosen safety margin, $d_critical$. In this situation there is no conflict detected for the ownship and it should follow its nominal trajectory. Conversely, when the separation between the intruder and the ownship is less than the safety margin, a penalty is applied in order to enforce the resolution of the conflict. This penalty value is larger as the intruder gets closer to the ownship.

Finally, as the main point the conflict resolution system must enforce is collision avoidance, a large negative penalty is applied if the ownship collides with one of the intruders.

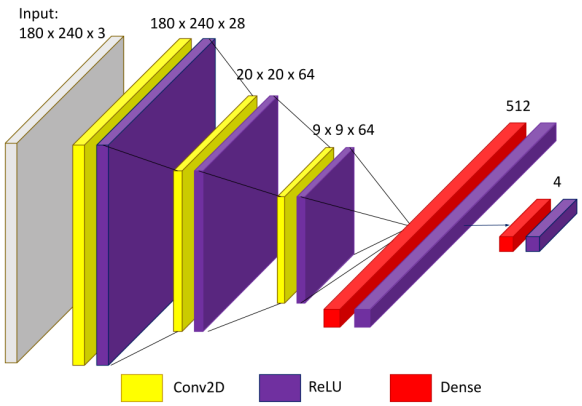

Figure 2. Scheme of the proposed architecture.

3.2.2 Creation of a DQN agent

Architecture

A scheme of the proposed architecture is illustrated in Fig. 2. The number of layers, number and size of filters has been extracted from a previous study on DQN applied to the Atari Pong game [8].

In total, we have implemented 3 convolutional layers (with filter sizes and strides of 8, 4, 3 and 4, 2, 1, respectively), each followed by a ReLU activation and two Fully-Connected layers, also followed by ReLUs. The last FC layer has an output size of 4, which corresponds to the number of possible actions, as the model will be outputting the Q-value associated to each of them.

Sub-sampling was achieved without pooling layerby making the convolutional kernels larger (ranging from 20x20 pixels to 7x7 pixels per kernel). This conserves the location of the features (e.g. the ball), which is vital for most games like Atari Pong and specially for the problem we want to solve in this Project. Adam optimizer was chosen, and the loss function is the Hubber loss. Tensorflow backend was employed for this implementation.

State definition

In our DQN structure, the states are directly extracted from the input images. Using a single input image as a state, only the current position of the elements could be inferred. In this problem however, it is desirable to also know the velocity and acceleration of each element (aircraft) involved in the problem. This can be achieved by establishing as a state a sequence of images comprising the current image and the three previous.

ϵ, for the ϵ-greedy policy

The policy we will set for our RL agent will be epsilon-greedy, meaning that part of the time the action to be taken will be chosen randomly to let the model learn new states (exploration) and the rest of the time the action chosen will be the one with larger score (exploitation).

$$a(t) = \begin{cases} \mathrm{maxQ_t(a)} & \text{probability} \quad 1-\epsilon \\ \mathrm{any(a)} & \text{probability} \quad \epsilon \end{cases} \tag{1}$$

The implementation of the ϵ-greedy policy in our code will include a decay factor for ϵ. This means that at the beginning of the computation, ϵ will be set to 1 (meaning that the model will behave in exploration mode). Then, after a certain threshold number of episodes, the model will start learning and the decay factor will be applied to decrease ϵ and promote exploitation as training advances.

Summary of hyperparameters

After tuning, the resulting hyperparameter values set to our model are gathered in Table 1.

Parameter	Description	Value
α	Learning rate	0.025
γ	Discount factor	0.95
ϵ_0	Initial ϵ	0.9
ϵ_{min}	Minimum ϵ	0.05
ϵ_{decay}	Decy value for ϵ	0.0009
β_1	Adam's 1st moment decay	0.9
β_2	Adam's 2nd moment decay	0.999

Table 1. Summary of the hyperparameters set to the CNN-DQN model.

4. Data

Most research efforts in the literature have majorly been to identify resolution maneuvers from historic data using different algorithms or classify the resolutions into categories. A major problem with these approaches is that historical data such as the ADS-B do not contain any potential flight conflict since Air Traffic Control Officers (ATCOs) already intervene and resolve such situations [4]. Moreover, it is difficult to identify individual ATCO strategies from the historical data. ATCOs dominantly rely on the spatial and temporal patterns of traffic rather than the instantaneous positions of the aircraft.

Due to this scarcity of real data to train and test our model, the input data employed comes from generated synthetic trajectories. The ownship position and heading are randomly initialized and the positions and trajectories of the intruders are created in such a way that there is at least one point of conflict with the ownship's nominal path.

5. Experiments and Evaluation

Performance metrics

We define the success rate as the percentage of episodes/simulations in which the ownship has reached the target and the conflict resolution rate (CRR) as the percentage of conflicts resolved within one episode.

Figure 3 illustrates the evolution of the success rate during training for four different scenarios, with 1, 2, 3 and 4 intruders, respectively. In all scenarios the final success rate converges to 100% but during the first 1000 episodes of training (Figure 3) is when the difference between the model performance for each scenario varies. It can be seen how the model converges to the maximum success rate at 1000 episodes for the 1-intruder scenario, as expected, because there was only one conflict halfaway of the ownship's trajectory, so it had space and time enough to recover the nominal trajectory and reach the target. On the other hand, scenarios with 2 and 3 intruders present a similar evolution of the SR at the beginning (300 episodes), but the rise decreases for the scenario with 3 intruders. As expected, the trend is to achieve slower learning of the model when the number of intruders increases.

If paying attention to the CRR, we can see an example of its evolution during training on Figure **??** for the 1-intruder scenario. This metric gives us a sense on how conflict resolution is enforced within the model: even though we want the ownship to reach the target, the system priority is to avoid an imminent collision and thus it is required that the CRR converges to 100%. It can be seen that this requirement is accomplished after approximately 1500 training episodes.

Comparison with a baseline DQN model

One of the purposes within this Project is to address the suitability of employing images as inputs to an air conflict detection and resolution problem, which was already debated in the literature [10].

To conduct a comparative study, a parallel DQN algorithm with a very similar structure than the presented in the Methodology section has been implemented within this Project. The custom OpenAI Gym will be essentially the same except that the state returned by the environment won't be the canvas with all aircrafts in it (and the ownship target) but an array containing the x-y positions of each aircraft (first, the ownship coordinates and then the corresponding to the intruders).

On the other hand, the architecture of the agent's model has been modifyied by removing the convolutional layers and keeping only the two fully connected ones. The state representation will be an array containing the states of the last 4 steps, so that velocity and acceleration of the elements could be extrapolated from the sequence, analogous to the CNN model.

Comparison of success rate evolution

It is desired to evaluate the convergence of the baseline DQN model in comparison with the proposed CNN-DQN architecture. Figure 5 illustrates the evolution of the success rate for both models for each scenario (1, 2, 3 and 4 intruders). It is observable that for 1 and 2 intruders (subfigures a) and b) respectively), the CNN-DQN model gets higher success rate values at a lower number of episodes, whereas increasing the number of intruders (subfigure d)) the trend is the opposite. For the case of 3 intruders the evolution of the success rate is similar. These results can give some insight on how the model performance can be affected by using either purely positional input data or using images. However, other features should be taken into account for comparison, such as the quality of predicted trajectories by each model.

Predicted trajectories

An interesting qualitative analysis that has been performed is based on the observation of the model's output trajectories. As mentioned before, there exist many possible combinations of actions leading to a same end, but due to the aircraft's physical and timing constraints, it is desired to get the optimal maneuver.

Figure 6 illustrates the resulting predicted trajectories using the scenarios corresponding to 2 and 3 intruders, respectively, and using the proposed CNN-DQN model and the DQN model without CNNs. The plotted trajectories are the ones rated with the highest score by the model during training. For case (b), with 3 intruders, the trajectories predicted with both methods are very close, whereas for case (a), with 2 intruders, they are dissimilar when handling conflicts. It can be seen that the prediction of the model with CNNs has less turns and leads to a more optimal solution than the model without CNNs. This result gives some insight of how using CNNs in this kind of problems can lead to a better performance than using purely position data as state representation.

6. Conclusion and Future Works

In this Project a DQN-based tool using CNNs has been developed to be applied to conflict detection and resolution between flying vehicles such as UAVs. A DQN algorithm, based on previous works on solving the Atari Pong game, has been developed. The DQN agent created for our problem application consists of a CNN-FC model that takes images as inputs. These input images are a synthetic representation of the trajectories for the different aircrafts involved in the conflict and this approach has been chosen due to its

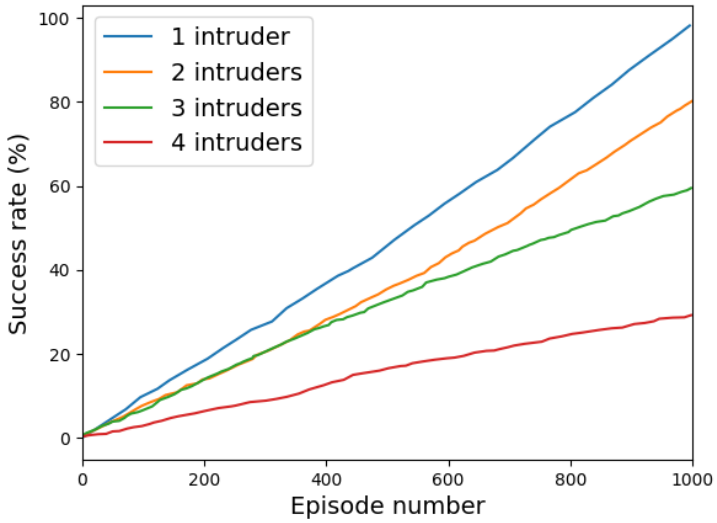

Figure 3. Evolution of the rate of success of the different scenarios during training.

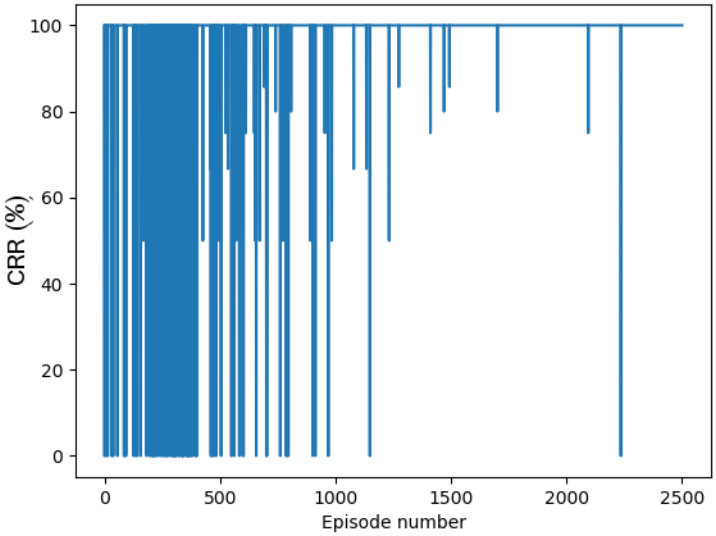

Figure 4. Evolution of the conflict resolution rate for the 1-intruder scenario.

suitability to handle the inclusion of multiple elements in the problem without altering the state space and take advantage of the employment of CNNs.

The performance of the proposed tool has been tested on different scenarios containing 1, 2, 3 or 4 intruders, respectively. The success rate (percentage of times the ownship reaches its target) and resolution rate (percentage of conflicts successfully solved) have been quantified for each scenario. On the other hand, a DQN algorithm analogous to the main one has been created without the employment of CNNs in order to assess the difference in performance between both methods given the increase in computational expense required by CNNs.

For future works, the current two-dimensional approach could be extended to the three dimensions in order to account for altitude-corrective actions, and also variations in velocity could be addressed. Also, the creation of wider scenarios with more vehicles involved should be addressed if a

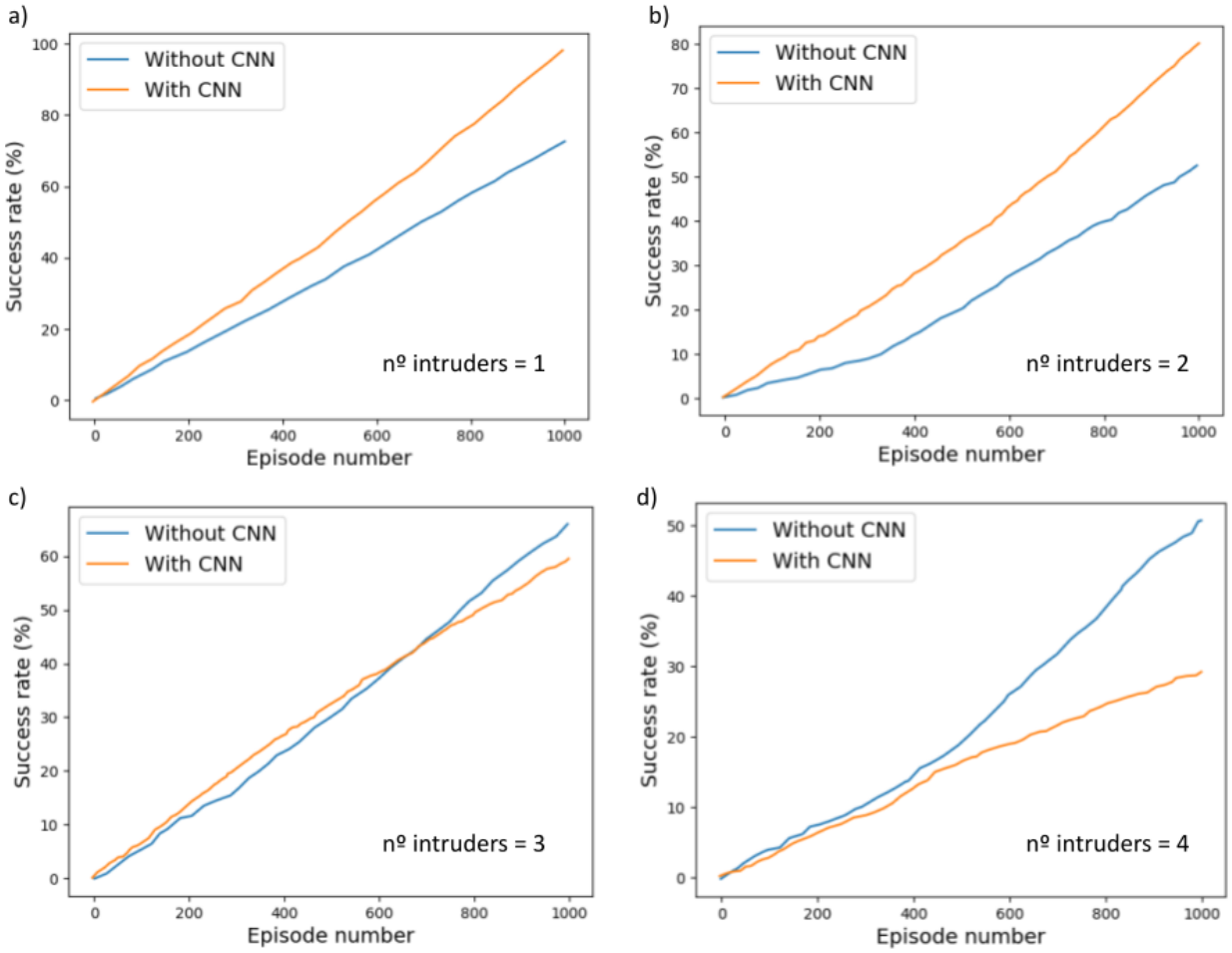

Figure 5. Comparison of the success rates achieved during training with the baseline DQN model and the CNN-DQN model for the scenarios with (a) one intruder, (b) two intruders, (c) three intruders, (d) four intruders.

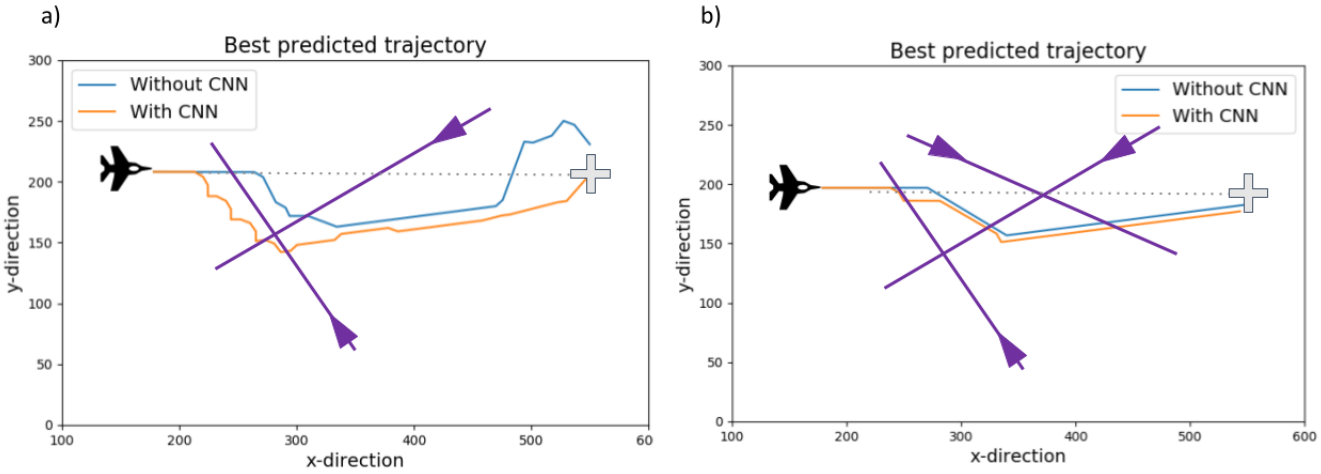

Figure 6. Trajectories predicted by the proposed model in the scenario with two intruders (a) and with three intruders (b). Purple lines indicate intruders' trajectories while the gray cross represents the target point the ownship shall reach.

powerful machine is employed. Finally, the image representation chosen for this Project does only contain the positions of the involved elements, but other representation methods (such as SSDs [11]) could provide more information to the model and therefore lead to more efficient and reallistic resolution maneuvers.

References

[1] Openai gym. https://gym.openai.com/. 2

[2] Marc Brittain and Peng Wei. Autonomous air traffic controller: A deep multi-agent reinforcement learning approach, 05 2019. 2

[3] Yunsheng Fan, Zhe Sun, and Guofeng Wang. A novel reinforcement learning collision avoidance algorithm for usvs based on maneuvering characteristics and colregs. *Sensors*, 22(6), 2022. 2

[4] Yash Guleria, Phu Tran, Duc-Thinh Pham, Nicolas Durand, and Sameer Alam. A machine learning framework for predicting atc conflict resolution strategies for conformal automation, 2021. 4

[5] Y.-X Han, Xinmin Tang, and Songchen Han. Conflict resolution model of optimal flight for fixation airway. *Jiaotong Yunshu Gongcheng Xuebao/Journal of Traffic and Transportation Engineering*, 12:115–120+126, 02 2012. 1

[6] International Civil Aviation Organization (ICAO). Icao circular 328 - unmanned aircraft systems (uas). *ICAO, Tech. Rep., 2011*. 1

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. 2

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. 4

[9] Óscar Pérez-Gil, Rafael Barea, Elena López-Guillén, Luis M. Bergasa, Pedro A. Revenga, Rodrigo Gutiérrez, and Alejandro Díaz. Dqn-based deep reinforcement learning for autonomous driving. In Luis M. Bergasa, Manuel Ocaña, Rafael Barea, Elena López-Guillén, and Pedro Revenga, editors, *Advances in Physical Agents II*, pages 60–76, Cham, 2021. Springer International Publishing. 2

[10] Md Siddiqur Rahman, Laurent Lapasset, and Josiane Mothe. Aircraft conflict resolution using convolutional neural network on trajectory image. In Ajith Abraham, Niketa Gandhi, Thomas Hanne, Tzung-Pei Hong, Tatiane Nogueira Rios, and Weiping Ding, editors, *Intelligent Systems Design and Applications*, pages 806–815, Cham, 2022. Springer International Publishing. 2, 5

[11] S. Rooijen, Joost Ellerbroek, Clark Borst, and Erik-Jan Van Kampen. Toward individual-sensitive automation for air traffic control using convolutional neural networks. *Journal of Air Transportation*, 28:1–9, 05 2020. 2, 8

[12] Hikaru Sasaki, Tadashi Horiuchi, and Satoru Kato. A study on vision-based mobile robot learning by deep q-network. In *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 799–804, 2017. 2

[13] Dong SUI, Weiping XU, and Kai ZHANG. Study on the resolution of multi-aircraft flight conflicts based on an idqn. *Chinese Journal of Aeronautics*, 35(2):195–213, 2022. 2, 3

[14] Jiang Xurui, Wu Minggong, Wen Xiangxi, Tu Congliang, and Wang Zibolin. A multi-aircraft conflict resolution method based on cooperative game. In *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 774–778, 2017. 1

[15] Jianya Yuan, Hongjian Wang, Honghan Zhang, Changjian Lin, Dan Yu, and Chengfeng Li. Auv obstacle avoidance planning based on deep reinforcement learning. *Journal of Marine Science and Engineering*, 9:1166, 10 2021. 2

[16] Li Ang Zhang, Jia Xu, Dara Gold, Jeff Hagen, Ajay K. Kochhar, Andrew J. Lohn, and Osonde A. Osoba. *Air Dominance Through Machine Learning: A Preliminary Exploration of Artificial Intelligence?Assisted Mission Planning*. RAND Corporation, Santa Monica, CA, 2020. 3

[17] Peng Zhao and Yongming Liu. Physics informed deep reinforcement learning for aircraft conflict resolution. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2021. 2