# Application of DRL-Based Algorithm for the Resolution of Strategic Conflicts in U-Space Airspaces

Manuel González [*], Sandra Amarillo [‡], Alex Sanchis [‡], Juan Vicente Balbastre [‡]

Posted Date: 2 March 2026

doi: 10.20944/preprints202603.0011.v1

Keywords: PX4; U-space; machine learning; strategic resolution; flight authorisation service

 doi:10.20944/preprints202603.0011.v1



*Article*

# Application of DRL-Based Algorithm for the Resolution of Strategic Conflicts in U-Space Airspaces †

**Manuel González *, Sandra Amarillo ‡, Alex Sanchis ‡ and Juan Vicente Balbastre ‡**

Air Navigation Systems Research Group, ITACA Institute, Universitat Politècnica de València, València, Spain
* Correspondence: mgonrod1@upv.edu.es
† Presented at the 15th EASN International Conference, in Madrid, Spain, from 14-17 of October 2025.
‡ These authors contributed equally to this work.

**Abstract**

The rapid expansion of Unmanned Aircraft Systems (UAS) operations has created an urgent need for scalable strategic conflict resolution methods within the U-space framework. When requested 4D flight plans overlap with previously authorised ones, the Flight Authorisation Service (FAS) denies the request and can provide the UAS operator with an alternative, conflict-free route. While traditional pathfinding algorithms ensure optimal routes, their computational cost creates a critical bottleneck during the flight activation phase or emergency missions, which demand near-instantaneous responses. To address this, we propose a three-stage framework. First, an Octree spatial partitioning discretises the airspace to identify occupied cells. Second, the A* algorithm is implemented to establish an optimal reference route. Finally, a standard Deep Reinforcement Learning (DRL) model, trained on realistic PX4 Simulator trajectories and using a well-adjusted reward function, generates alternative paths that optimise distance and energy. Our results demonstrate that this DRL architecture achieves near-optimal routing behaviour. Crucially, it reduces computation time by several orders of magnitude compared to A*, solving complex conflicts in milliseconds rather than seconds. We conclude that a simple, well-tuned DRL architecture overcomes latency limitations of classical pathfinding while achieving optimal results, ensuring rapid, safe, and efficient conflict resolution for high-density U-space.

**Keywords:** PX4; U-space; machine learning; strategic resolution; flight authorisation service

## 1. Introduction

Although Unmanned Aircraft Systems (UAS) have been used in military operations since the early 20th century — including World War I [1] and the Vietnam War [2] — widespread interest in their civil applications did not emerge until the beginning of the 21st century. Since then, the technology has experienced rapid growth, driven by expectations of numerous professional applications across diverse sectors such as urban mobility, parcel delivery, public safety and security, entertainment, agriculture, and infrastructure inspection, in addition to recreational use. Most of these operations are envisaged in th Very-Low-Level airspce (below 500 ft) over urbarn areas beyond the visual line-of-sight (BVLOS) from the remote pilot. This type of operations poses a challenge to civil aviation authorities, since they are not covered by visual flight rules (VFR) and instrumental flight rules (IFR) used in manned aviation [3].

In this context, the concept of UAS Traffic Management (UTM) emerged in the mid-2010s as a dedicated component of air traffic management aimed at enabling the safe, efficient, and economically sustainable integration of UAS operations. According to the International Civil Aviation Organization (ICAO) [4], UTM comprises the set of services and operational functions required to ensure the seamless integration of UAS into the airspace, encompassing both airborne and ground-based elements.

The European implementation of UTM, established through Commission Implementing Regulation (EU) 2021/664 on a regulatory framework for the U-space [5], introduces the concept of U-space

 doi:10.20944/preprints202603.0011.v1



airspace, defined as a UAS geographical zone within which UAS operations are permitted only through the use of a predefined set of U-space services. These services include, at a minimum, network identification, flight authorisation, geo-awareness, and traffic information services. In addition, specific geographical zones in which UAS operations may be prohibited or restricted can be designated within a U-space airspace.

The Flight Authorisation Service (FAS) constitutes a core safety mechanism for UAS operations within U-space airspace. UAS operators submit Flight Authorisation Requests (FARs) to the FAS, each including the intended UAS trajectory represented as a sequence of four-dimensional (4D) volumes that account for flight execution uncertainties up to a 95% probability bound. Upon receipt, the FAS applies predefined spatial and temporal deviation thresholds, introducing a safety buffer that supports an appropriate balance between risk mitigation and efficient airspace utilisation. The FAS subsequently assesses whether the 4D trajectory associated with the newly submitted FAR generates any spatial or temporal intersections with previously authorised 4D trajectories and verifies compliance with applicable geographical restrictions, including prohibited UAS geographical zones. If no conflicts are identified and airspace capacity constraints are respected, the request is authorised. In the event that an overlap is detected, the FAS may propose an alternative conflict-free 4D trajectory to the UAS operator [5].

FARs with the same priority level are processed on a first-come, first-served basis. However, high-priority FARs may override previously authorised lower-priority requests. Consequently, the introduction of high-priority operations can significantly affect already approved trajectories. To mitigate this impact, the FAS may propose a 4D trajectory for the high-priority operation that minimises disruption to authorised flights while still satisfying the mission-specific priority requirements. While standard UAS operations are typically planned well in advance of take-off, priority missions—such as the urgent delivery of medical supplies—require near-immediate authorisation.

In all cases, once a FAR has been approved by the FAS, the UAS operator is required to request activation of the authorisation shortly before take-off. According to the European regulatory framework [6], the time elapsed between the activation request and the confirmation of flight activation shall not exceed 5 seconds in 95% of cases. Prior to issuing the activation, the FAS performs a final pre-departure verification to ensure that the authorised 4D trajectory remains compatible with other ongoing operations and with applicable airspace constraints.

One of the fundamental safety mechanisms in U-space airspace is the segregation of manned aircraft operating under Air Traffic Control (ATC) instructions through Dynamic Airspace Reconfiguration (DAR). A DAR entails the temporary deactivation of part of, or the entirety of, a U-space airspace. It is triggered when the relevant Air Traffic Services (ATS) unit determines that a manned aircraft under ATC control is about to enter the U-space airspace. Upon activation of a DAR, the FAS shall reject any FAR containing 4D trajectories that intersect the affected airspace volume, withdraw previously issued flight authorisations impacted by the DAR, and update the trajectories of affected airborne UAS to ensure that they vacate the impacted volume and either safely complete their mission or perform a controlled landing.

The strategic deconfliction process carried out by the FAS is not posed as a continuous-time optimal control problem, but as an intersection-free allocation of 4D airspace volumes associated with flight authorisations and UAS geographical zones, as stated in [5]. Therefore, the key computational aim is efficient 4D occupancy management and collision queries rather than continuous trajectory integration.

*Standard Specification for UAS Traffic Management (UTM) UAS Service Supplier (USS)* Interoperability, *ASTM F3548-21* also reinforces the idea of the discretisation of the airspace. It proposes that the implementation of the airspace representation can be conceptualised as a grid, and mapping a flight into the airspace representation determines which grid cells it intersects [7].

In this context, the projected growth in demand for Unmanned Aircraft Systems (UAS) across multiple operational domains has generated an urgent need to effectively manage the emerging U-

 doi:10.20944/preprints202603.0011.v1



space airspace in order to ensure safety, scalability, and operational efficiency. To address this challenge, research efforts are being conducted worldwide, particularly at the intersection of two rapidly evolving technological fields: UAS and Artificial Intelligence (AI) [8]

This work follows a three-stage pipeline: First, the tool generates a 4D graphical representation of the occupied airspace, enabling operators to visually identify candidate routes. Second, an A* (A-star) pathfinding algorithm is applied in the discretised space to obtain a conflict-free reference trajectory. Third, a Reinforcement Learning (RL) approach is introduced as an alternative to compute near-optimal routes at lower computational cost, and compared with the A* reference trajectory, as an indicator of its performance.

Having said that, this work proposes an Octree-based discretisation of the U-space in cells, as Amarillo, S. *et al.* proposed [9]. It provides an efficient way to manage airspace representation in accordance with the aforementioned standard *ASTM F3548-21*.

To ensure that the learning-based planner is trained and evaluated under operationally meaningful conditions, the proposed framework relies on realistic trajectory data rather than purely geometric or kinematic toy models. Training a DRL agent on straight-line segments with artificial noise may lead to policies that perform well in simplified settings but fail to capture the behaviour of real UAS operations, where the executed path is shaped by autopilot logic, vehicle dynamics, and environmental disturbances.

Therefore, we generate realistic UAS trajectories using a PX4 Software-In-The-Loop (SITL) pipeline coupled with Gazebo, which emulates autopilot-driven flight behaviour and enables the systematic injection of operational disturbances and off-nominal events (e.g., wind effects, link disruptions and failsafe behaviours) [10]. This approach provides higher-fidelity datasets for U-space systems assessment and increases the credibility and potential generalisation of the learned policy when compared to trajectories built from idealised motion assumptions. In addition, the use of fast-time simulation enables large-scale dataset generation across heterogeneous scenarios, alleviating the current lack of widely adopted performance models for UAS trajectories (i.e., an analogue of BADA for UAS).

As for 2D or 3D resolution, we adopt a hierarchical planning strategy: a primary 2D planner operates at cruise altitude, consistent with common UAS operational profiles, since they are usually planned to reach cruise altitude and maintain it until descent, thereby maintaining energetically efficient trajectories. On the other hand, a secondary 3D planner is triggered only when no feasible 2D solution exists due to airspace congestion. Provided that 3D resolution can infer energetically costly movements, the proposed 3D model is evaluated not only in terms of distance but also in terms of energy. The 2D resolution is usually more efficient in terms of energy and computational efficiency. This two-layer design prioritises fast, energy-efficient planning while preserving completeness when 2D becomes infeasible.

However, the use of DRL is undoubtedly questionable. Although it is possible to achieve a near-optimal result (compared to A*) while reducing computation time, on some occasions, it may not be extremely decisive. In time, two different stages coexist when it comes to flight planning:

First, the approval stage. In this part, flight plans are requested for approval. Although EASA mentions in *Easy Access Rules for U-space* [6] that the time required to approve a flight authorisation request should be as short as practically possible, this stage typically takes place from hours to days before the flight. Provided that a typical resolution might take up to seconds in A* and centiseconds in DRL, there may be no tangible differences. Of course, every second matters, but not as importantly as in the next stage.

Secondly, the activation stage comes into play. The flight's activation occurs at the exact moment it starts. In the time between the previous and the present stage, the state of the U-space airspace is likely to change, e.g., new authorisations, new restricted geographical zones, and changing wind conditions. In this stage, a nearly-immediate response is required (*Standard F3548-21* states that the activation of a flight request should be made within $n$ seconds in 95% of the time [7] and EASA, in

 doi:10.20944/preprints202603.0011.v1



*Easy Access Rules for U-space* interpretes that it should be made, e.g., within 5 seconds [6]); therefore, DRL outperforms A* in this aspect and is essential to maintain U-space airspace efficiency [6].

Furthermore, if a high-priority request arrives, e.g., an emergency mission, those flights that interfere with the emergency flight shall be cancelled. However, if the DRL tool can provide a free-of-conflict route instantly, there is no need to cancel the previously accepted flights, and airspace efficiency would not be impacted.

Finally, the proposed framework is evaluated in terms of feasibility, optimality with respect to the A* reference, computational latency and (in the 3D fallback) energy-related cost. Results show that the Reinforcement Learning-based planner can deliver absolutely optimal deconflicted routes (in terms of cells) while substantially reducing response time, which is particularly relevant for the activation stage under dynamically changing U-space conditions.

The structure of this paper follows a systematic approach, beginning with a *Literature Review* section (Section 2) that presents recent, high-impact work in this field. Next, the *Materials and Methods* section (Section 3) describes in detail the resources and procedures used throughout the study, including Airspace Discretisation, UAS Geographical Zones, the energy model, and the operation of the A* (reference route) and RL. Afterwards, the *Deep Reinforcement Learning Models* section (Section 4) outlines the design choices and training workflow of the proposed RL models. Then, the *Results* section (Section 5) reports and analyses the performance of the models in comparison with A*, followed by the *Discussion* section (Section 6), where the feasibility of the tool is analysed. Then, the *Conclusions* (Section 7) highlight the main contributions and future directions of this work. To close, the *Future Work* section (Section 8) mentions some improvements or alternatives that might be worth researching in the future.

## 2. Literature Review

The rapid proliferation of Unmanned Aerial Systems (UAS) and the imminent integration of these vehicles into urban environments (Urban Air Mobility, UAM) have driven a transformation in airspace management. Forecasts indicate that the traffic volume of UAS in low-altitude airspace will be much higher than the historical density of commercial aviation [11], necessitating a change in the paradigm of airspace control, which has historically been centred on humans. To ensure the U-space's scalability is not compromised, it appears necessary to digitalise U-space services. More specifically, this paper pivots around the Strategic Conflict Resolution Service, which offers an alternative optimal route when a flight plan has been withdrawn due to an overlap with a previously accepted flight or a restricted spatial volume.

Traditionally, in path planning, classic algorithms have been the most widely used. Graph-based algorithms such as A* or Dijkstra approach the airspace as a discrete grid and evaluate the transitions between nodes, accumulating costs and applying heuristic functions [12]. A* algorithm, despite being widely considered as the golden standard to guarantee global optimality in terms of distance, suffers from an asymptotical computational complexity, which in the worst case, grows as $\mathcal{O}(b^d)$, where $b$ represents the branching factor and $d$ the length of the route (see Equation 4). This exponential growth suggests that for extensive flight routes that cross long, significant urban areas, the computational time required exceeds the time limit for the activation-phase path planning.

However, in these extensive data environments, not only must the solving method be fast enough, but the management of the discretised airspace should also be efficient enough to prevent discretisation from becoming the resolution bottleneck. Thus, the scalability of this path-planning resolution strongly relies on efficient spatial indexing and filtering. Conventional geometric comparison methods, which scale at $O(n\hat{}2)$, are being replaced by three-dimensional Morton (Z-curve) and Octree spatial discretisation. In 2026, Li, Q. *et al.*, in *Research on Multi-Airspace Conflict Detection and Resolution Method Based on Octree Spatial Grid Partitioning* proposed an efficient approach to provide reliable solutions based on Octree [13]. Furthermore, in 2025, Amarillo, S. *et al.*, in *Proposal of UAS Strategic Conflict Detection Concept with a Centralised Service in Multi-USSP Environment Using an Octree Data*

 doi:10.20944/preprints202603.0011.v1



*Structure* [9], presented an algorithm that aligns with the standard ASTM F3548-21 [14], and successfully demonstrated the impact of the developed solution. This approach has been adopted to discretise the U-space airspace in this work.

As for the integration of Artificial Intelligence in U-space, more specifically, Machine Learning models, advances have been made. In 2024, Gordo, V. *et al.*, in *Feasibility of Conflict Prediction of Drone Trajectories by Means of Machine Learning Techniques* [15] demonstrated that integrating high-density U-space structures with ML classifiers, such as Neural Networks (NN), allows for the prediction of collisions up to 100 seconds in advance with high precision, showing the advantages of using ML in terms of computational time in U-space, paving the way for the integration of this technology in Strategic Conflict Detection and Resolution.

More specifically, in Conflict Resolution, although the A* algorithm remains a benchmark for global optimality, its computational complexity grows exponentially (see Equation 4), leading to significant latency issues. Further advances have been made in order to make the most of it, e.g., in 2024, He, Y. *et al.*, in *A new method for unmanned aerial vehicle path planning in complex environments* [16] developed an improved A* for UAS Route Planning, enhancing its computational efficiency.

The current research not only focuses on improving classical algorithms, but also on finding suitable and robust alternatives. Even quantum technologies are starting to be used. Innan, N. *et al.* developed a Quantum-Assisted Path Planner for UAS Navigation, comparing its performance with A*, showing promising results [17].

However, the main alternative technologies being researched and the most promising are AI-based. The research community is transitioning from classical search-based algorithms to Reinforcement Learning (RL) to address the "computational complexity explosion" in pathfinding across complex, extensive 3D volumes. RL is one of the favourites, and Q-Learning is one of the most widely used RL algorithms for this purpose. Nevertheless, it has difficulties generalising in large state spaces. In 2013, Mnih, V. *et al.*, in *Playing Atari with Deep Reinforcement Learning* [18] proposed combining Q-Learning with Deep Neural Networks (Deep Q Network - *DQN*) to address the aforementioned problem in discrete environments, creating a reference technique that is widely used in this day and age. In a 2D discrete environment, in 2017, Wang. Z *et al.*, in *Deep Reinforcement Learning Assisted UAV Path Planning Relying on Cumulative Reward Mode and Region Segmentation* [19], proposed a Deep Reinforcement Learning (DRL) Model for UAS Path Planning, based on DQN, showing its robustness in solving path planning problems in discrete environments, along with its computational advantages in comparison with A*. In a more recent work, in 2023, Westheider, J. *et al.*, in *Multi-UAV Adaptive Path Planning Using Deep Reinforcement Learning* [20], empirically demonstrated that DRL can outperform state-of-the-art non-learning-based methods. Furthermore, in 2025, Sharma, G. *et al.*, in *Deep Reinforcement Learning-Based Framework for Path Planning of AUAVs* [21], presented a Soft Actor-Critic (SAC) method and demonstrated its feasibility for 2D path planning.

Complex DRL architectures, such as D3QN and PPO, exist and have proven successful in UAS path planning. E.g., in 2025, Jarray, R. et. al., in *Improved deep Q-network-based UAV path planning in partially observable environments* [22], proposed an innovative variant based on Improved DQN for partially observable scenarios. Despite all these advances, recent literature suggests that the performance bottleneck in RL path planning is often not the architecture of the whole model itself, but rather the optimisation of hyperparameters (e.g., Learning Rate) and the design of the reward signal. In 2024, Dierkes, J. *et al.*, in *Combining Automated Optimisation of Hyperparameters and Reward Shape* [23] empirically demonstrated that the learning rate (which is one of the most complex hyperparameters) and the reward shape are mutually dependent, and that a standard RL agent with finely tuned hyperparameters can sharply outperform more complex models that lack this optimisation. Having said that, in this work, a standard DQN architecture has been chosen, achieving outstanding optimality results compared to A*.

 doi:10.20944/preprints202603.0011.v1



## 3. Materials and Methods

### *3.1. UAS Trajectories Generation and Fidelity*

The accurate assessment of U-space services requires high-fidelity trajectory data. The aviation industry currently lacks a standardised performance model for Unmanned Aircraft Systems (UAS) comparable to the Base of Aircraft Data (BADA) used in manned aviation. Furthermore, large-scale empirical datasets of autonomous drone flights in urban environments are virtually nonexistent. Consequently, trajectory modelling often relies on simplified kinematic approaches, such as Uniform Rectilinear Motion (MRU) models appended with random directional noise. These basic models fail to capture the complex physical realities of UAS flight, omitting critical factors like multirotor inertia, aerodynamic drag, environmental disturbances, and continuous autopilot corrections. Recent studies emphasise that multirotors operating in low-altitude urban environments are highly susceptible to wind gusts, which cause significant deviations from the intended flight path that simple kinematic models cannot predict [24].

To overcome these limitations, in this study we employ a high-fidelity Software-in-the-Loop (SITL) simulation environment utilising the PX4 open-source flight control software paired with the Gazebo physics engine [10]. This approach effectively bridges the simulation-to-reality gap by executing the exact flight control stack, guidance laws, and navigation algorithms that run on physical hardware. Thus, the resulting trajectories represent physically constrained flight executions rather than idealised geometric paths.

To illustrate the necessity of this high-fidelity approach, Figure 1 demonstrates the temporal evolution of spatial deviation between two identical flight plans executed under different environmental conditions (nominal versus constant wind). Unlike simplified MRU models that inject basic statistical noise, the error introduced in the PX4 SITL environment is the direct result of complex physical interactions.

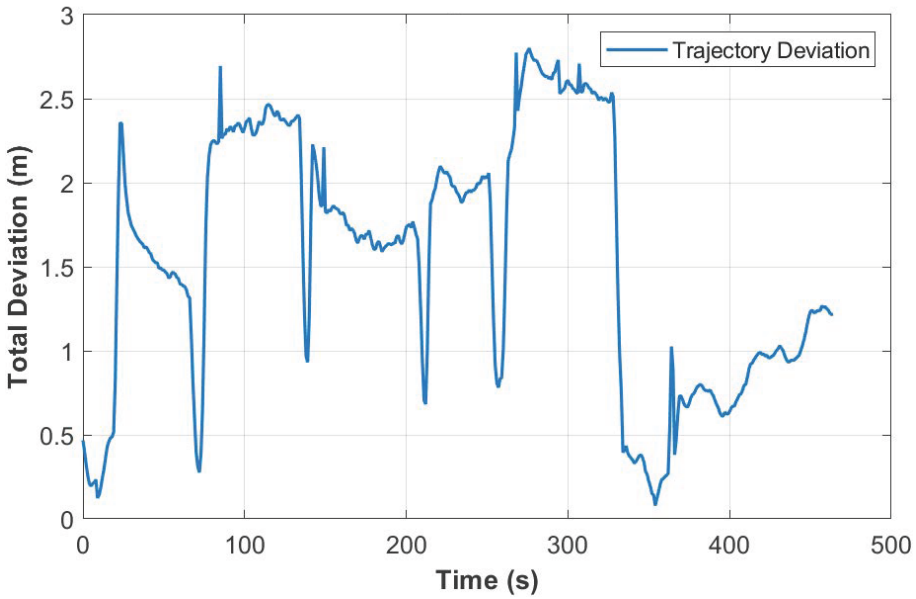



**Figure 1.** Temporal evolution of spatial deviation between identical flight plans executed under nominal and wind-affected conditions in the PX4 SITL environment [10].

As observed in Figure 1, the deviation exhibits an increasing trend during the cruise phase, punctuated by sharp, periodic corrections as the autopilot actively compensates upon reaching specific waypoints. The most significant irregularities manifest during transitional flight phases. The takeoff phase shows a steep initial deviation as wind resistance counters the vertical ascent, while the landing phase produces the peak deviation due to lateral drift induced during a gradual descent. Comparing these physically constrained trajectories to a basic MRU model would reveal massive positional

 doi:10.20944/preprints202603.0011.v1



discrepancies, rendering simplified kinematic models inadequate for rigorous U-space regulatory validation.

For this study, we generated a baseline dataset of ten straight cruise trajectories subjected to nominal default wind conditions. This setup forces the PX4 autopilot to actively compensate for aerodynamic drift, creating realistic micro-deviations and speed fluctuations. We subsequently use these baseline executions as the structural foundation to build the pseudo-random trajectories evaluated later in this work. Generating these trajectories requires a robust automation infrastructure. We developed a Python-based toolchain leveraging the MAVSDK library to autonomously upload flight plans and execute missions. To resolve native PX4 UDP port allocation bottlenecks and ensure scalability, we encapsulated the entire simulation environment within Docker containers. This containerised architecture isolates each process, preventing cross-talk and bypassing port conflicts. Consequently, the system dynamically allocates resources to run multiple trajectory simulations concurrently, drastically reducing computation time while continuously exporting high-resolution telemetry data (3D position, velocity, and precise orientation) to a central database.

### *3.2. Airspace Discretisation and Cell Representation*

As mentioned in the previous paragraphs, the trajectory generator provides a realistic, dynamic representation of UAS motion that accounts for a wide range of interfering factors.

However, the European Regulation and the *ASTM F3548-21 Standard* establish that the airspace representation should be conceptualised as a grid, and that both flight trajectories and restricted volumes (e.g., Geozones, obstacles, etc.) are mapped into those grid cells. These cells consist of 4D volumes, which are 3D volumes extended with a temporal dimension, defined by start and end times. The start and end times account for the time span during which the flight is located within one cell [7].

To enhance computational efficiency, we have adopted an Octree spatial partitioning to discretise the airspace into adaptive cells. This hierarchical data structure recursively subdivides the 3D space into eight octants (children nodes). As illustrated in Figure 2, only the octants containing relevant geometry are further refined, while empty regions remain as larger, undivided cells. In this example, the refinement process focuses on the yellow sub-cells, representing the maximum subdivision resolution of the structure, whereas the blue and red nodes represent intermediate levels of the hierarchy.

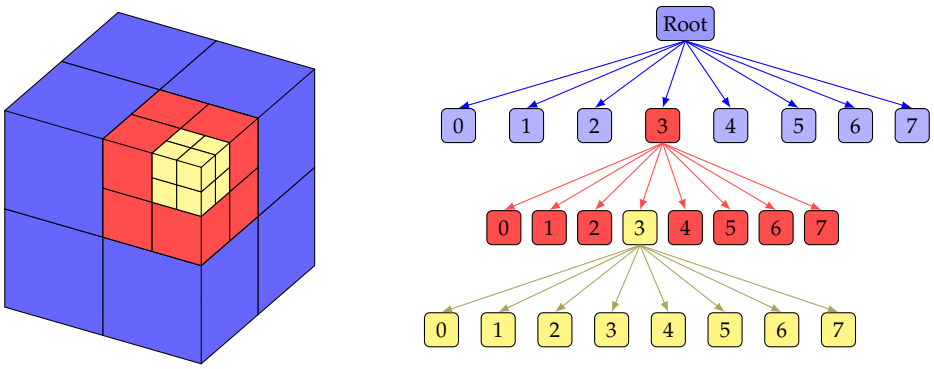



**Figure 2.** Octree structure representation.

Therefore, as shown in Figure 2, the action of locating a cell is performed by descending the tree from the root to a cell, with a depth $d$ bounded by the refinement level. For a balanced tree, this yields a logarithmic access cost,

$$T_{\text{loc}} = \mathcal{O}(d) = \mathcal{O}(\log N), \tag{1}$$

where $N$ denotes the number of cells.

The computational difference between a conventional and an Octree partitioning is higher as the resolution increases, i.e., as the size of the cells reduces, provided that the computational cost of a conventional partitioning is,

 doi:10.20944/preprints202603.0011.v1



$$T_{\text{loc}} = \mathcal{O}(N), \tag{2}$$

After justifying the use of this technique, a specific approach was developed by Amarillo, S. *et al.* in *Proposal of UAS Strategic Conflict Detection Concept with a Centralised Service in Multi-USSP Environment Using an Octree Data Structure* [9]. They proposed an implementation central to the Search For Overlap Service (SFOS), a supporting microservice within the Flight Authorisation Service (FAS) framework. By utilising the octree's recursive subdivision, the SFOS can efficiently build an airspace representation and map 4D operational intents into specific grid cells to identify spatial overlaps (temporal overlap is detected as an extension; it is an added dimension to the cell). This approach not only facilitates interoperability in multi-USSP environments by allowing different providers to declare conflicts based on shared information (per *Standard ASTM F3548-21* [14]), but also leverages parallel computing to significantly accelerate the deconfliction process as traffic density increases.

### *3.3. UAS Geographical Zones*

The *UAS Geographical Zones*, commonly referred to as *Geozones*, represent a fundamental tool for airspace management within the U-space ecosystem. According to *Commission Implementing Regulation (EU) 2021/664* [5] and *2019/947* [25], these are defined as portions of airspace where UAS operations are facilitated, restricted, or excluded to address risks related to safety, security, privacy, or environmental protection.

From a dynamic management perspective, as outlined in the EASA *Easy Access Rules for U-space* [6], Geozones are not merely static constraints but can be activated temporarily to accommodate specific aerial events or emergency situations. An illustrative example occurs during an aerial exhibition: a *Geozone* is defined by the event's horizontal limits and extends vertically from the surface to the upper limit of the U-space volume. This temporary segregation effectively mitigates the risk of Mid-Air Collisions (MAC) by preventing unauthorised UAS from entering the high-risk volume.

In terms of computational modelling, representing these volumes accurately is critical for strategic deconfliction. While standard flight operational intents usually occupy one-dimensional lines (in terms of cells), Geozones frequently take the form of large, semi-static polyhedral volumes. As explained in Subsection 4.1, the geometric disparity between a standard flight path and a massive Geozone makes the last the most restrictive, and it requires a specialised Reinforcement Learning (RL) approach able to handle these scenarios. By utilising an Octree-based spatial discretisation, our model can efficiently handle the "empty" space within these large volumes.

### *3.4. Conflict Detection Workflow*

The U-space standardisation framework establishes a list of services that should be mandatory for the safe integration of Unmanned Aircraft Systems (UAS) in complex environments. According to *Commision Implementing Regulation (UE) 2021/664* [5] and the *EASA Easy Access Rules for U-space* [6]. One of them is the Flight Authorisation Service (FAS), which plays a critical role in flight permissions, ensuring that every 4D trajectory does not overlap with other occupied volumes or Geozones, neither in volume nor in time. According to the regulatory framework, the service that provides an alternative route if the flight is rejected (this tool) is optional.

Therefore, as portrayed in Figure 3, a flight plan, containing a 4D trajetory, is read by the FAS. The FAS will check whether there is an overlap with another cell that contains a previously accepted flight during the timespan when the flight is scheduled to take place. If the FAS doesn't detect an overlap, it will most probably validate the flight plan. Otherwise, if it detects an overlap, it will withdraw the flight plan. If a flight plan is withdrawn, the developed DRL-based tool is used to find an alternative route that avoids occupied cells while optimising distance (and energy in the 3D model).

 doi:10.20944/preprints202603.0011.v1



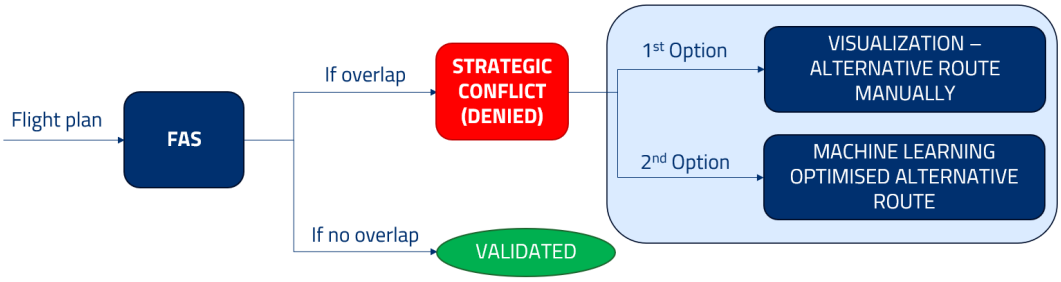



**Figure 3.** Workflow scheme.

In this work, the FAS notifies the DRL tool when a flight has been withdrawn. Then, this tool obtains the 4D trajectory of the withdrawn flight and the occupied cells from the Octree. All these cells (in LLA) are mapped into an indexed plane (i, j) - also k in 3D, provided that the DRL model performs much better in these kinds of environments.

### *3.5. Energy Model*

An approximate energy model has been developed to estimate the relative energy efficiency of the 3D planner. This model is also embedded inside the *Reward Function* of the RL agent so that, besides reaching the goal conflict-free, it explicitly optimises energetic cost. Note that the 2D model does not include this term, since its action space is restricted to constant-altitude motion and therefore excludes the most energy-demanding behaviours associated with vertical transitions.

The purpose of the model is to assign a *normalised* dimensionless energetic penalty to each discrete manoeuvre. Costs are defined relative to a straight, level step (cost equal to 1), and are derived by combining power and energy-per-distance figures reported by Gong, H. *et al.* in *Modelling Power Consumptions for Multi-rotor UAVs* [26], combined with the results from the work of Jacewicz, M. *et al.* in *Quadrotor model for energy consumption analysis* [27]. Here, energy consumption in discrete manoeuvres is derived from continuous motion analysis. Provided that detailed manoeuvre-resolved consumption data are scarce, the model is a first-order approximation and assumes a multirotor (quadrotor) platform. This assumption is fairly reasonable in the present context, as the ENAC report indicates that at least 85% of UAS operators in Italy use quadrotors [28].

The following simplifications are adopted: wind and aerodynamic interactions are neglected; battery mass is assumed constant; acceleration effects are considered only for turning and vertical transitions; all manoeuvres are assumed feasible; and hover or fixed-wing dynamics are not modelled.

The resulting normalised costs (dimensionless, relative to a straight segment) are reported in Table 1. As expected, pure vertical ascent has a comparatively low cost due to the limited altitude increment per cell, whereas compound actions involving simultaneous turning and climbing (e.g., double ascending 180° turns) yield the largest penalties.

 doi:10.20944/preprints202603.0011.v1



**Table 1.** Final normalized energy costs for all Maneuvers.

| Section | Maneuver | Cost | Maneuver | Cost |
|---|---|---|---|---|
| Basics | Straight segment | 1.0000 | Ascent moving[1] | 1.0012 |
| | Vertical ascent | 0.0488 | Double ascent moving[2] | 2.0024 |
| | Vertical descent | 0.0323 | Descent moving | 1.0005 |
| | Double descent moving | 2.0010 | | |
| Diagonal Movements | Straight segment | 1.4142 | Ascent | 1.4151 |
| | Double ascent | 2.8302 | Descent | 1.4146 |
| | Double descent | 2.8292 | | |
| 45° turns | Horizontal | 1.4616 | Descending turn | 1.3467 |
| | Double descending | 2.6934 | Ascending turn | 1.3474 |
| | Double ascending | 2.6948 | | |
| 90° turns | Horizontal | 1.9233 | Descending turn | 1.6931 |
| | Double descending | 3.3862 | Ascending turn | 1.6939 |
| | Double ascending | 3.3878 | | |
| 135° turns | Horizontal | 2.8849 | Descending turn | 2.5396 |
| | Double descending | 5.0792 | Ascending turn | 2.5408 |
| | Double ascending | 5.0816 | | |
| 180° turns | Horizontal | 4.3273 | Descending turn | 3.8094 |
| | Double descending | 7.6188 | Ascending turn | 3.8112 |
| | Double ascending | 7.6224 | | |

[1] Moving: it is being done while moving horizontally as well. [2] Double ascent: case when the drone is going downwards in movement and starts going upwards in movement. It is simplified to ascent twice.

### *3.6. Reference Method: A* Pathfinding*

A* is a classical graph-based pathfinding algorithm that extends Dijkstra's method with a heuristic function, making the search more efficient. In Equation 3, its mathematics are explained [29].

$$f(n) = g(n) + h(n), \tag{3}$$

where

- $g(n)$ is the accumulated cost from the initial node to the current node $n$.
- $h(n)$ is the heuristic function. Estimates the remaining cost from the current node $n$ to the goal. It should always underestimate the remaining distance; hence, the Euclidean distance is often used here.
- $f(n)$ is the total cost of the most efficient path from the initial point to the goal while travelling through $n$.

Under certain conditions that are met in this work, this algorithm ensures the most efficient (absolute-optimal) path. By default, it is used for distance optimisation. However, the cost functions can be modified to optimise the desired variables. In this work, both distance and energy are optimised, so the energy costs are included in the cost function.

However, these graph-based classical algorithms, despite being the most precise approach, incur extremely high computational costs, especially for long routes and 3D environments. Equation 4 shows that the computational complexity (proportional to cost) grows exponentially with the distance of the route, in the worst cases [29].

$$O(b^d), \tag{4}$$

where

- $b$ is the branching factor (number of adjacent nodes to the current node), which is 26 in the 3D environment ($3^3 - 1$), and 8 in the 2D environment ($3^2 - 1$).

 doi:10.20944/preprints202603.0011.v1



- $d$ is the route length, which quickly becomes prohibitive for long trajectories.

This is the main flaw of the algorithm, and the objective of this work is to develop an RL model that performs similarly to A* (ideally equal) and finds paths at a fraction of the computational cost.

### 3.7. Proposed Alternative: RL Model

As an alternative, we use Deep Q-Learning (DQN), which is a form of Reinforcement Learning (RL) that combines Q-learning with deep neural networks (DNN) to handle large state spaces [30,31].

The computational complexity gives the computation time of an RL model, as shown in Equation 5, where $d$ is the route distance, indicating that time grows linearly with $d$.

$$O(d), \tag{5}$$

The models have been created with the *stable-baselines3* library by OpenAI. These models base their learning on an *Agent* that learns from a *training process*, and it basically learns to perform based on a *Reward Function*. From the *state space* it sees, it predicts the expected *reward* and from the *Action Space* it chooses the action that is likely to provide the highest *reward* signal, further maximizing it (see Figure 4).

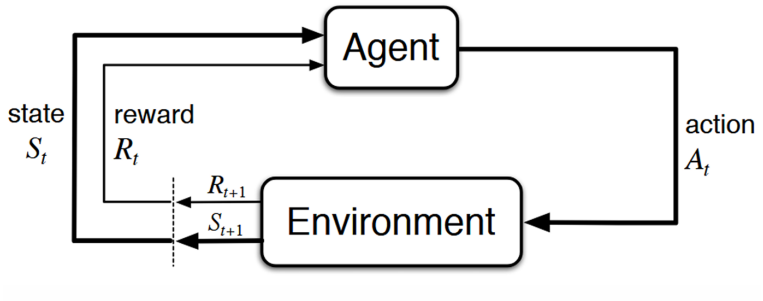



**Figure 4.** Reinforcement Learning Model scheme [30].

There are many different techniques for training an RL model, as well as an extensive set of advanced architectures. However, all of them base their learning on the same idea. The *Agent* is the component that makes the decisions, based on the *States* it receives (the states can be understood as the senses of the agent), and the expected *Reward* from taking an *Action*. E.g., if the *Agent* is learning to play football, it will take an action, e.g., kicking the ball, and if the ball goes into the goal, it will see it in the *States* and receive a positive *Reward*. Otherwise, it will receive a negative *Reward*. This way, through many episodes of training, the *Agent* will learn to predict the *Reward* signal after taking an *Action* in a determined *State*, and consequently, it will learn to maximise the *Reward*. So, if it has been programmed correctly, this results in the *Agent* learning to play football [30].

#### 3.7.1. Mathematical Formulation of the RL Approach

The routing problem is modelled as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, R, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $p(s', r \mid s, a)$ the transition probability, $R$ the reward signal, and $\gamma \in [0, 1)$ the discount factor [30].

At each discrete time step $t$, the agent observes the state $S_t \in \mathcal{S}$, selects an action $A_t \in \mathcal{A}$ according to a policy $\pi(a|s)$, receives a reward $R_{t+1}$, and transitions to a new state $S_{t+1}$. The objective is to maximise the expected discounted return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{6}$$

The optimal action-value function is defined as:

 doi:10.20944/preprints202603.0011.v1



$$q^*(s,a) = \max_{\pi} q^{\pi}(s,a), \tag{7}$$

and satisfies the Bellman optimality equation:

$$q^*(s,a) = \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \max_{a'} q^*(s',a')\right]. \tag{8}$$

Given the dimensionality of the discretised 3D U-space grid, a tabular representation of $q(s,a)$ is computationally impractical. Therefore, a Deep Q-Network (DQN) is employed to approximate the optimal action-value function, such that $q(s,a;\theta) \approx q^*(s,a)$, where $\theta$ denotes the neural network parameters. The parameters are updated by minimising the temporal-difference loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{(S_t,A_t,R_{t+1},S_{t+1})\sim\mathcal{D}}\left[\left(R_{t+1} + \gamma \max_{a'} q(S_{t+1},a';\theta^-) - q(S_t,A_t;\theta)\right)^2\right], \tag{9}$$

where $\mathcal{D}$ denotes the replay buffer distribution and $\theta^-$ the parameters of a periodically updated target network.

In the proposed framework, $S_t$ encodes the agent position within the discretised grid and its relation to the goal, while the transition structure is implicitly determined by the grid topology and occupied cells. The reward function embeds collision avoidance constraints and efficiency criteria (distance and energy cost), and $\gamma$ regulates the balance between short-term manoeuvre penalties and long-term path optimality.

Unlike A*, which relies on an explicitly defined heuristic function $h(n)$, the RL agent learns an implicit approximation of the cost-to-go function through interaction with the environment. Once trained, policy inference scales linearly with the number of executed steps, which becomes advantageous in high-resolution 3D U-space environments.

Two different DQN models have been proposed, following an exhaustive evaluation and the identification of the optimal RL hyperparameters and reward signal (since they are the most critical parts of the model [23]), as follows.

#### 3.7.2. Generalisation, Underfitting and Overfitting Phenomena

As mentioned before, an RL model learns from rewards obtained after taking actions in a defined space, which it perceives as its observation space. In other words, it can be viewed as a mapping from a dataset $D_{set}$ of learning samples $(x,y)$ into a predictive model $f(x|D_{set})$ [32].

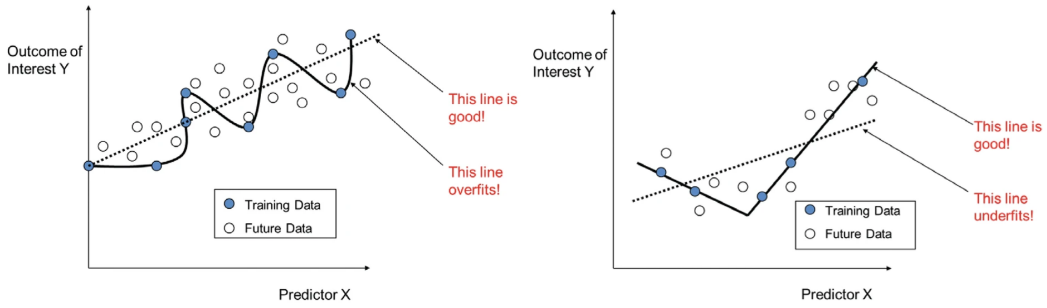
In 2024, Aliferis *et al.*, in *Overfitting, Underfitting and General Model Overconfidence and Under-Performance Pitfalls and Best Practices in Machine Learning and AI*, provided useful, intuitive explanations of these concepts. Figure 5 show a classic demonstration of a regression of a continuous outcome $Y$ given a continuous input $X$. Blue circles are training data, and white circles are unseen data [33].

In Figure 5a, the complex model (represented by the solid curve) fits the training data perfectly but fails to generalize to the unseen data. This phenomenon is known as overfitting: the model learns the noise in the training set rather than the underlying pattern. Conversely, the simpler model (dotted line) shows higher error on the training data but generalizes better to unseen data.

In Figure 5b, the complex model fits both the training and unseen data effectively, representing an optimal balance. Conversely, the simpler model (dotted line) fails to capture the data's structure, performing poorly on both sets. This is known as underfitting.

 doi:10.20944/preprints202603.0011.v1






(**a**) (**b**)

**Figure 5.** (**a**) Overfitting example. (**b**) Underfitting example [33,34].

When training an DRL model, excessive complexity often leads to overfitting, where the model memorises the training data but fails to generalise on unseen instances. On the other hand, an overly simplistic dataset can lead to underfitting, failing to capture the data's underlying structure and yielding low predictive accuracy across datasets. These two phenomena, over- and under-fitting, are directly related to the complexity and distribution of the training data.

## 4. Deep Reinforcement Learning Models

Two different approaches have been adopted to create a DRL Model that finds the optimal route. First, a 2D model has been created and trained. As mentioned in Section 1, 2D resolution will be prior to 3D, given that it is generally more efficient both computationally and energetically. If no path is found in 2D or if the mission strictly requires 3D motion (not usual), the 3D resolution will take place. For that purpose, a 3D model has been designed.

### *4.1. DQN Model - 2D*

As explained in Subsection 3.3, in some specific cases, where the flight takes place at a constant altitude (as in most of cases), and the strategic conflict takes place with a *Geozone* instead of a Flight, there is no point in designing and training a 3D model, since it is not necessary, and it is further more complicated than a 2D model, due to the number of possible actions to choose and the number of states to process. Furthermore, in this model, energy will not be taken into account, since it has proven to work very efficiently without it. When it comes to RL, simpler is better, so we avoid unnecessary actions and states.

#### 4.1.1. State Space - 2D

Here, the state space is designed differently, covering values from -1 to 1:

- *Current Direction*: 2 elements, being -1, 0, or 1.
- *Vision Vector*: 24 elements. Contains information about the two adjacent cells in all directions, being 0 if the cell is free and 1 if there is an obstacle.
- *Relative Distance*: 2 elements. Contains the normalised distance from the current position to the goal. It is normalised by the grid size, taking values of 1 or -1 if the distance is maximum, and 0 if it is in the same cell, with continuous values (32-bit precision) otherwise.

This consists of the state space of a single frame. However, in this training, the *frame stacking* technique has been used. It consists of saving the states of the $N$ previous steps. This way, we give the agent short-term memory and prevent it from getting stuck in loops. In this case, the entire state space consisted of the current state and the 3 previous states. In total, the size of the state space is:

$$28 \frac{\text{states}}{\text{frame}} \times 4 \text{ frames} = 112 \text{ states} \tag{10}$$

 doi:10.20944/preprints202603.0011.v1



#### 4.1.2. Reward Function - 2D

The importance of designing an accurate reward function has already been mentioned previously. Furthermore, prestigious research supports the idea that keeping the reward function simple is better. In 2017, Silver, D. *et al.* presented the famous *AlphaGo Zero* [35]. There, the new model (trained only with +1 when victorious and -1 when defeated) is compared against its predecessor, AlphaGo (trained with complex rewards - the first model to beat a world champion -), showing superiority, winning 100-0.

Then, the reward function is presented as follows:

$$R = \begin{cases} R_{col}, & \text{if there is a collision} \\ R_{goal}, & \text{if goal achieved} \\ R_{dist} + R_{step}, & \text{otherwise} \end{cases} \tag{11}$$

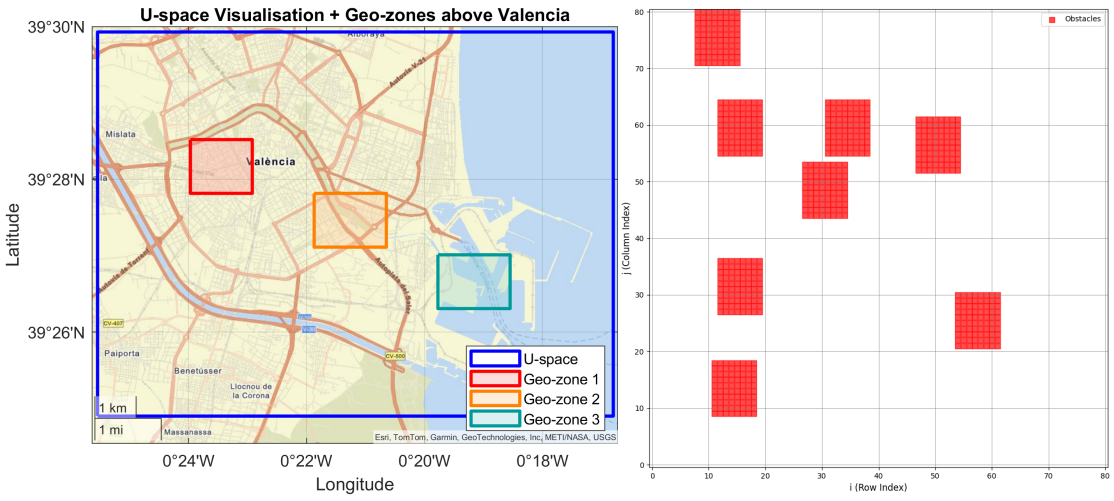
where

- $R_{col} = -5$ is the terminal reward for collisioning,
- $R_{goal} = +5$ is the terminal reward for achieving the goal,
- $R_{dist} = d_{prev} - d_{new}$ is the distance reward, positive when the agent gets nearer to the goal and negative otherwise,
- $R_{step} = -0.1$ slight penalisation for each step taken.

A notable change compared to the 3D reward function is its simplicity. In the previous model, the energy was taken into account. However, as mentioned before, here the simplicity was paramount and, as can be seen in the Training Results of this model, the performance has been much better.

#### 4.1.3. Training Process - 2D

Again, the model has been trained using synthetic data created from trajectories generated by the aforementioned PX4 SITL Simulator [10]. However, in this training, not only flight trajectories have been included, but also Geozones, explained in Subsection 3.3. The shapes of both the training set and the realistic geozones can be noted in Figure 6. Figure 6a portrays a real scenario where a flight plan is being blocked by the presence of three different geozones in a row, it is a similar case to the Scenario 2 of the evaluation. On the other hand, Figure 6b show the shape of one of the grids with which the model has been trained. As it can be noted, the training scenarios are more complex, to ensure a good performance in real scenarios.




(**a**) (**b**)

**Figure 6.** (**a**) Realistic geozones scenario. (**b**) Geozones of the training set.

 doi:10.20944/preprints202603.0011.v1



Finally, the model has been trained over approximately 25,000 episodes across 300 different grids of varying shapes, with start and end points generated randomly, again successfully avoiding *overfitting*.

#### 4.1.4. Training Results - 2D

Here, the training results of the 2D model are presented. Although the training has been much shorter than the 3D model (25,000 episodes vs the 100,000 of the previous one), the results show better learning, thanks to the simplicity and robustness of the reward function and the state space.

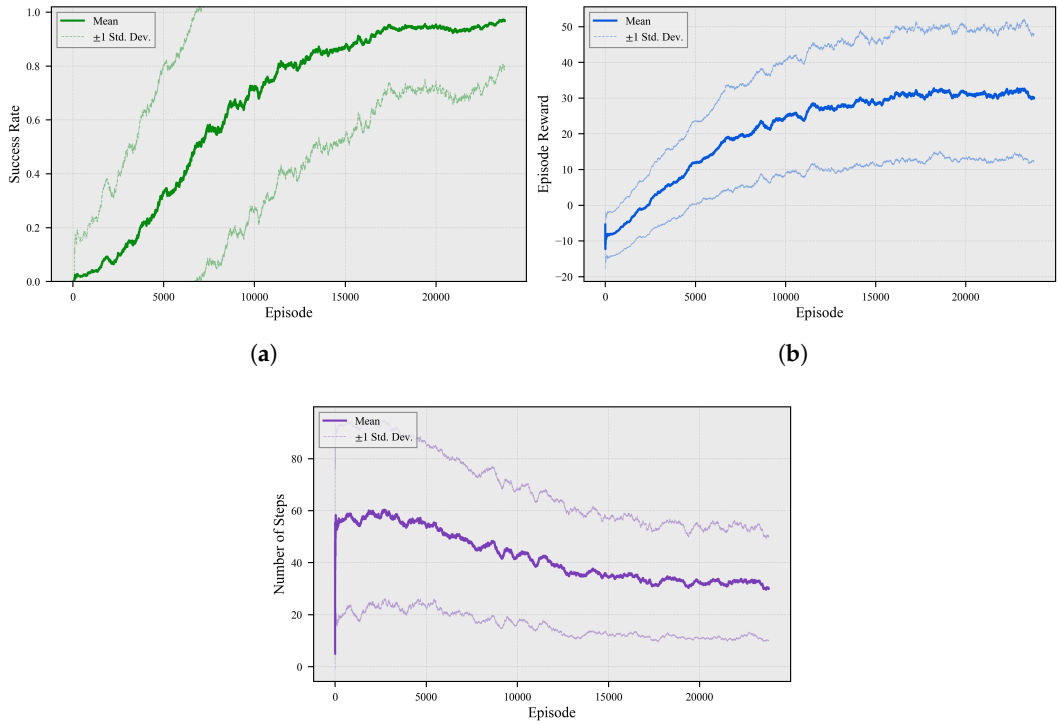



**Figure 7.** (**a**) Success rate 2D. (**b**) Mean of episode rewards 2D. (**c**) Number of steps 2D.

First of all, in Figure 7a, an increase from 0 to an average of 95% in the Success Rate is observed. First, the model searched without any information, and it properly learned to predict the highest reward, achieving the goal in the vast majority of cases.

Figure 7b shows the mean Reward obtained throughout the training. A tendency to go up from the very beginning can be observed. Again, this plot indicates that the model has learned to maximise its Reward and, consequently, if it has been programmed correctly, to achieve the goal in a near-optimal way.

Finally, Figure 7c shows the number of steps taken per episode. First, the average is 60, and decreases to around 30. This means the model has learned to achieve the goal along shorter paths, resulting in near-optimal behaviour.

One single plot doesn't really demonstrate that the model has learned properly. E.g., it can learn to achieve the goal (success rate near 100%), but due to a bad reward function, it may run many unnecessary loops or take sub-optimal paths. However, after analysing the three of them, we can confidently predict that this model is robust.

### *4.2. DQN Model - 3D*

#### 4.2.1. State space - 3D

The state space of the 3D Model is as follows, covering values from -1 to 1:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2026 doi:10.20944/preprints202603.0011.v1



- *Current Direction*: 3 elements, being -1, 0 or +1.
- *Vision Vector*: 26 elements. It is the same as 2D, but only information about one adjacent cell in all directions.
- *Relative distance*: 3 elements. It is the same as 2D, but in the 3D plane (one more element).
- *Euclidean Normalised Distance*: 1 element.

In total, for a single training frame, the agent will process 32 different states. In this case, it is not necessary to apply the technique of *frame stacking*, provided that the agent has more freedom to move and is less likely to get into loops.

#### 4.2.2. Reward Function - 3D

The reward function is, by far, one of the most critical parts of the model, since it determines its final behaviour. It can be regarded as what shapes the model's behaviour. See Equation 12.

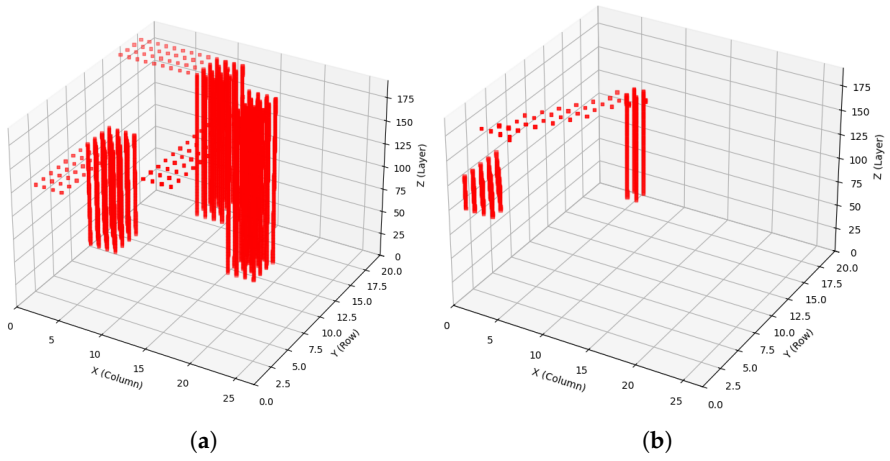
$$R = \alpha_{dist} \cdot R_{dist} + \alpha_{en} \cdot R_{en} + \alpha_{step} \cdot R_{step} + R_{term}, \tag{12}$$

where

- $R_{dist} = d_{prev} - d_{next}$ (distance improvement, positive if it gets nearer the goal),
- $R_{en}$ penalises energetically inefficient actions (according to the energy model - See Section 3.5),
- $R_{step}$ is a fixed step cost,
- $R_{term}$ is the terminal reward ($+10$ goal, $-50$ collision/out),
- $\alpha_{dist}, \alpha_{en}, \alpha_{step}$ are the corresponding weights, adjusted accurately.

#### 4.2.3. Training Process - 3D

The model has been trained using synthetic data created from realistic flight trajectories generated by the PX4 SITL Simulator [10]. These have been taken as a reference to synthetically create new trajectories pseudo-randomly, with the same shape and structure. The similarity between the simulator and the pseudo-random trajectories can be seen in Figure 8. The reason why they have been created pseudo-randomly is to ensure the generalization of the RL model [36]. As it can be noted, the pseudo-randomly generated trajectories have the same shape as the other ones but contain a higher obstacles density. This is intentional, since if the model performs well in **(b)**, it will definitely perform well or even better in the real one, **(a)**.




(**a**) (**b**)

**Figure 8.** (**a**) Pseudo-randomly generated trajectories. (**b**) PX4 Simulator trajectories.

The training process has been carried out through 100,000 episodes, throughout 100 different scenarios, with start and end points generated randomly at the beginning of each episode, again ensuring the generalization of the model.

 doi:10.20944/preprints202603.0011.v1



#### 4.2.4. Training Results - 3D

First, the results of the training process are presented in Figure 9.

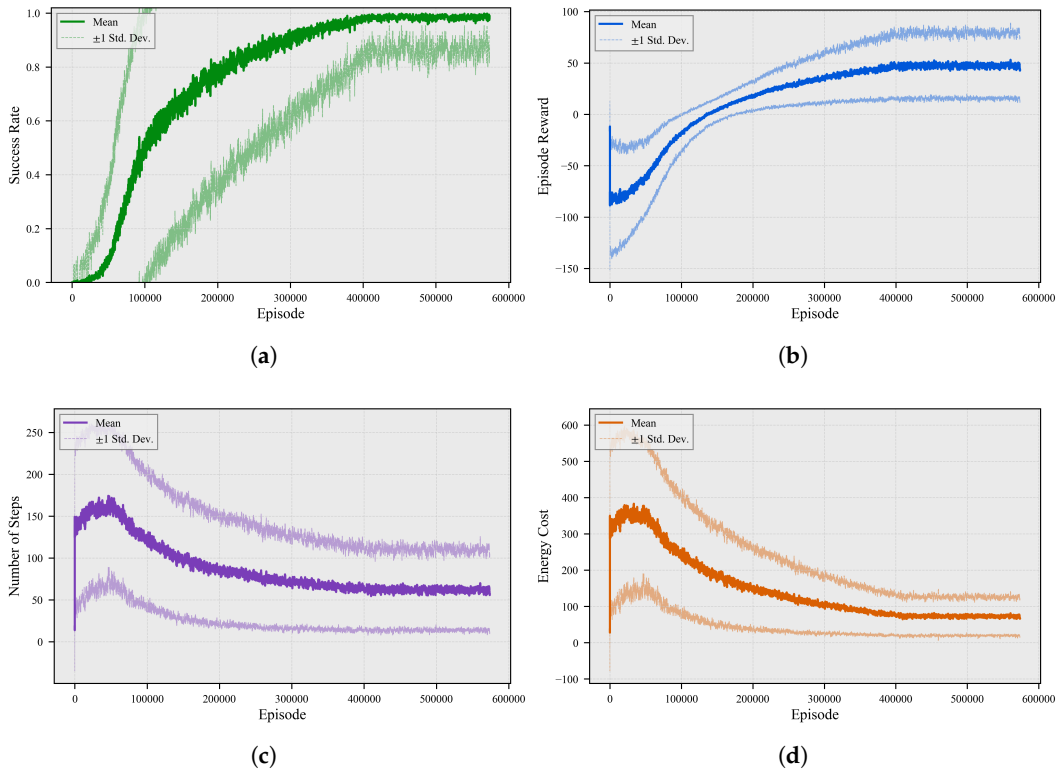



**Figure 9.** (**a**) Success rate. (**b**) Mean of episode rewards. (**c**) Number of steps. (**d**) Energy costs.

In Figure 9a, the *success rate* is shown, improving from 0 at the beginning of the training and ending up at around 95%, meaning that on 95% of the occasions, the agent is able to successfully reach the goal without colliding. This shows that the agent learns to achieve the goal, but it doesn't reveal how it does so.

The *episode rewards*, portrayed in Figure 9b, show an improvement from $-80$ to around $+50$. The absolute value of the reward tends to increase along the training. This means that the agent has learned to maximise its reward, and, consequently, if the *reward function* has been programmed properly, it has learned to successfully optimise distance and energy.

Figure 9c shows that the number of steps has significantly decreased during training, which indicates that the model is able to find routes in a smaller number of iterations. This plot, combined with the previous ones, clearly indicates that the model has learned to efficiently achieve the goal.

Furthermore, Figure 9d shows that the energy costs strongly decrease from the beginning to the end, confirming that the model is very likely to have adopted the desired behaviour. Please note the similarity between this plot and Figure 9c, which derives from the fact that the energy costs and the steps taken are directly related to each other.

## 5. Results

### *5.1. 2D Model*

The 2D Model will be assessed not only with the presence of flights, but also with the restriction of geozones. These kinds of volumes extend through an extensive area in the horizontal plane. Extremely importantly, none of these scenarios has been included in the training dataset, meaning that if it performs well, the model has *Generalised* correctly and avoided *Overfitting*. Provided that these geozones can be strongly prohibitive, e.g., a geozone on an area where a military base is located, a

 doi:10.20944/preprints202603.0011.v1



protection buffer of 5 cells (approximately 200-250 m) is applied to all of them (cells in light blue in Figure 11 and Figure 13). Here, the evaluation of the model in three different scenarios is presented:

#### 5.1.1. Scenario 1 - 2D

The first scenario, as shown in Figure 10, presents a strategic conflict between a flight request and two UAS Geographical Zones (Geozones) that restrict operations within the affected volume. In this scenario, as well as in the following Geozones-based scenario, a safety buffer of 5 cells (approximately 250 m in the horizontal plane) is applied around each Geozone to account for their prohibitive nature and to ensure that no UAS trajectory intersects these restricted areas.

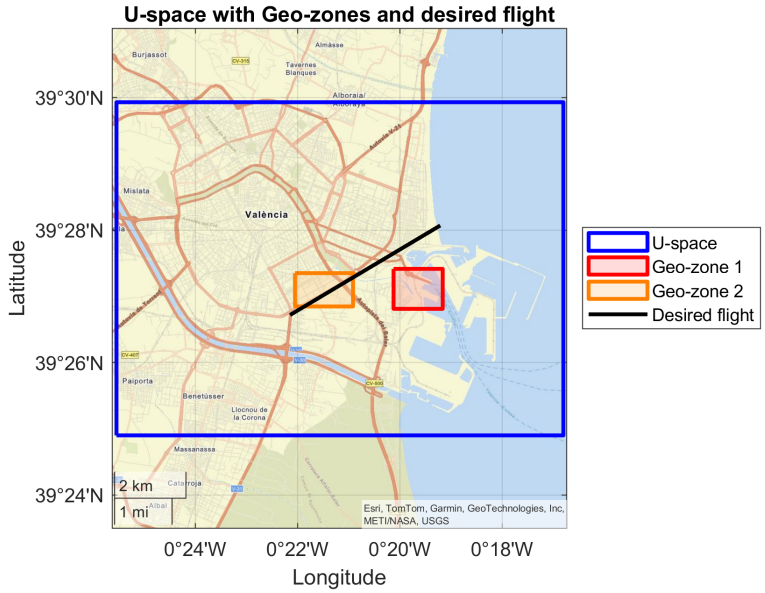



**Figure 10.** Representation of the flight request that is in conflict with two Geozones when it is about to cross a part of the city of Valencia.

In the first scenario, as can be seen in Table 2, ML (DRL) achieves a solid 100% optimality while reducing computation time in $\times 2.74$. This means that, with even better performance that A* (without introducing energy elements in the reward function), the model achieves smoother routes, with exactly the same length (it is impossible to achieve shorter routes since A* ensures optimality), in less time, allowing the resolution of more than two strategic conflicts by the time when A* could only solve one. Then, simplicity has been shown to be more efficient than a complex reward function, as predicted in Subsubsection 4.1.2.

Here, it is important to note that, in order to achieve the same level of time computation difference as the one achieved in the 3D case, the routes should be much longer. As Equations 4 and 5 show, the computational time of A* depends on the branching factor, while the computational time of RL keeps following the same linear tendency. In 3D, the computational complexity is $\mathcal{O}(26^d)$, while in 2D, it's $\mathcal{O}(8^d)$. This explains why, in 3D, the divergence happens too fast. However, the reduction is still extremely notable and worthy.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2026 doi:10.20944/preprints202603.0011.v1



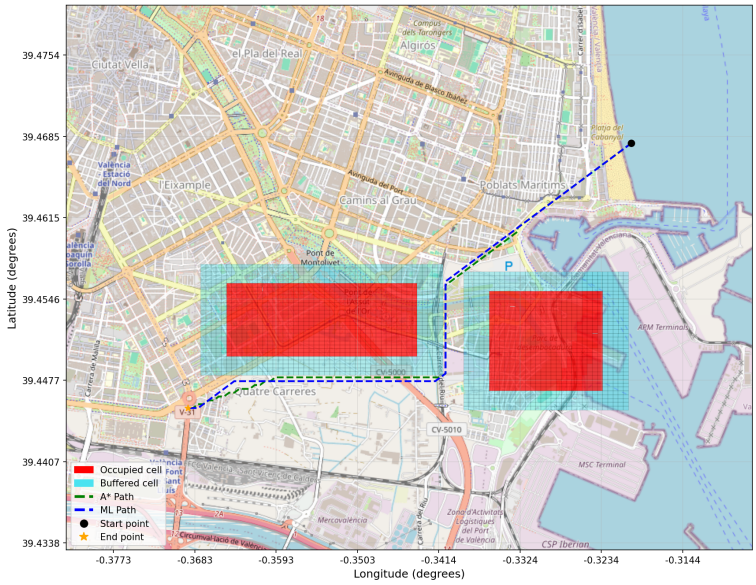



**Figure 11.** Evaluation of the 2D model in the first scenario.

**Table 2.** Numerical results of evaluation in the first scenario - 2D.

| Method | Length [steps] | Time [s] |
|---|---|---|
| A* | 111 | 0.0792 |
| ML (DRL) | 111 | 0.0289 |
| Difference | 0% | ×2.74 |

5.1.2. Scenario 2 - 2D

This scenario represents a city-scale conflict in the U-space located in Valencia. Figure 12 shows that the flight request completely overlaps with three different Geozones, provided that it crosses the entire city of Valencia.

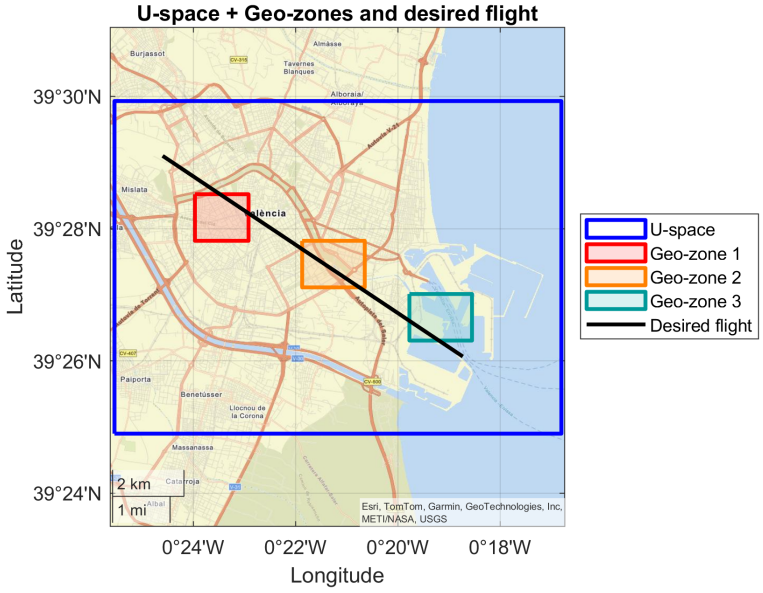



**Figure 12.** Representation of the flight request that is in conflict with three Geozones when it is about to cross the entire city of Valencia.

In this scenario, the more complex one, in which the model needs to solve a strategic conflict across three different geozones in the same U-space, the computation time has been reduced by up to a

 doi:10.20944/preprints202603.0011.v1



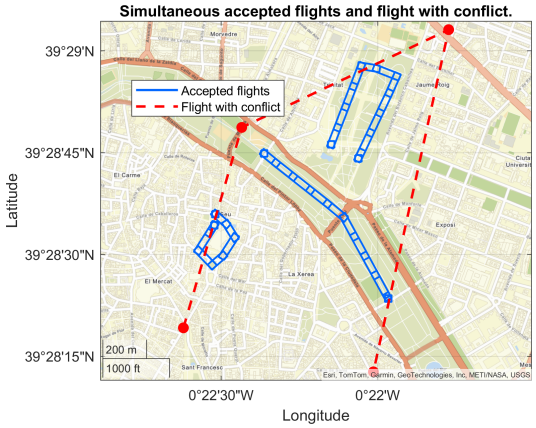



**Figure 14.** Representation of the operational volumes of the previously accepted flights (blue volumes) and the flight request that is in conflict (red line).

fourth, while achieving 99, 47% accuracy in terms of the optimal route length. This means that, with an absolute optimality in precision, the DRL model is capable of solving around four strategic conflicts by the time A* solves only one.

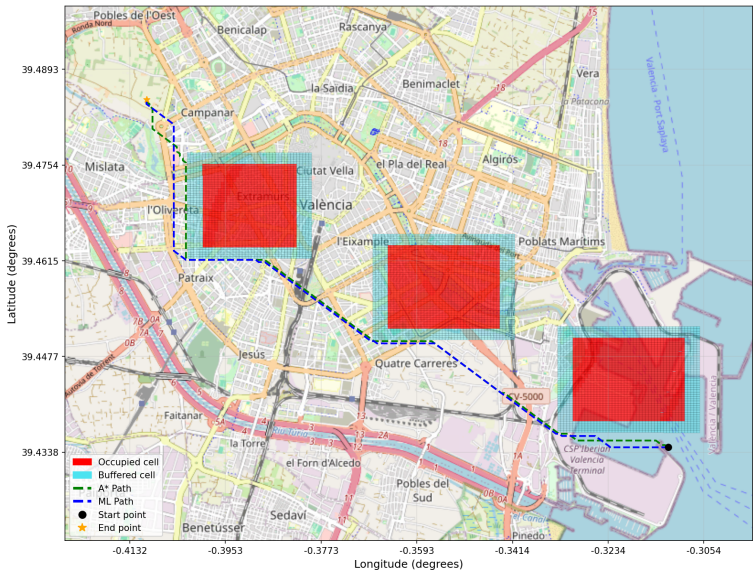



**Figure 13.** Evaluation of the model in the second scenario - 2D.

**Table 3.** Numerical results of evaluation in the second scenario - 2D.

| Method | Length [steps] | Time [s] |
|---|---|---|
| A* | 230 | 0.2484 |
| ML (DRL) | 231 | 0.0610 |
| Difference | 0.43% | ×4.072 |

5.1.3. Scenario 3 - 2D

In this scenario, the 2D model is assessed for the conflict with flights. The DRL model solves a flight plan with four waypoints (three segments). As shown in Figure 14, the flight plan request can't be approved because it is in conflict with the operational volumes of the previously accepted flight. If this were an emergency mission, three other flight plans would need to be cancelled. This is the impact of the tool.

Figure 15 shows a successful resolution when each segment is in conflict with three different previously accepted flights. The resolution is smooth, and as shown in the numerical results in Table 4,

 doi:10.20944/preprints202603.0011.v1



a 98.87% optimality is achieved, while reducing computational time by $\times 3.388$, showing the robustness of the DRL model.

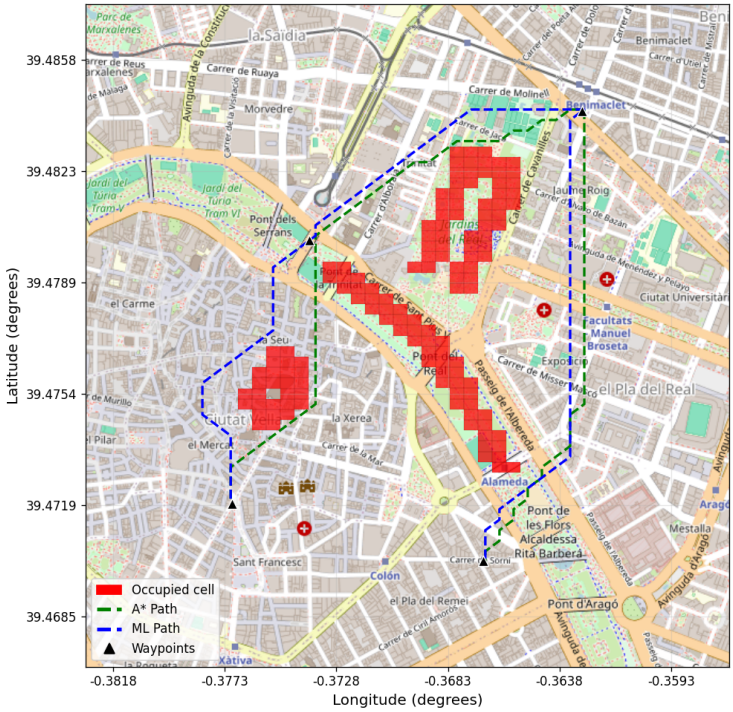



**Figure 15.** Evaluation of the model in the third scenario - 2D.

**Table 4.** Numerical results of evaluation in the third scenario - 2D.

| Method | Length [steps] | Time [s] |
|---|---|---|
| A* | 88 | 0.0820 |
| ML (DRL) | 89 | 0.0242 |
| Difference | 1.13% | $\times 3.388$ |

### *5.2. 3D Model*

Finally, the trained model has been evaluated in different environments generated by the aforementioned PX4 Simulator [10]. Importantly, none of these environments were part of the training dataset, meaning that the agent had no prior exposure to them. This methodology ensures that the results reflect genuine generalisation rather than memorisation, providing a realistic and trustworthy evaluation of the model's performance.

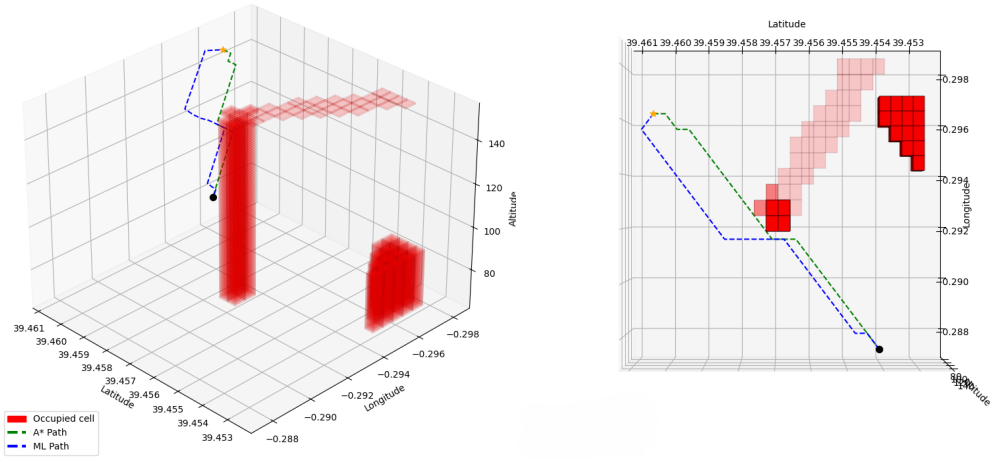
#### 5.2.1. Scenario 1 - 3D

First of all, please note that these 3D scenarios are extracted from the scenario where Amarillo, S. *et al.*, in *Proposal of UAS Strategic Conflict Detection concept with a centralised service in multi-USSP environment using an octree data structure* [9] simulated a scenario with 10 flights in the city of Valencia, simulating a U-space.

The first scenario evaluation is shown in Figure 16. The blue line shows the A* route while the green line shows the DRL one. Both of them reach the goal while avoiding the red (occupied) cells. Graphically, both are quite similar. In numerical results, shown in Table 5, it can be noted that the length and the energy increments 15% and 6.51% while the computation time is reduced to half of it. These results are considered near-optimal, since the increment is low yet worthy.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 March 2026 doi:10.20944/preprints202603.0011.v1





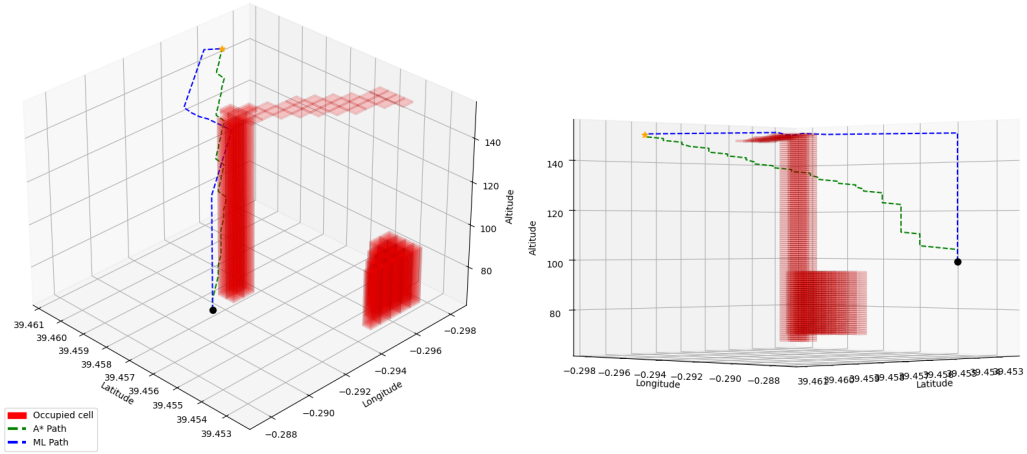

(**a**) (**b**)

**Figure 16.** Evaluation of the 3D model in the first scenario: (**a**) Perspective 1. (**b**) Perspective 2.

**Table 5.** Numerical results of evaluation in the first scenario - 3D.

| Method | Length [steps] | Energy [-] | Time [s] |
|---|---|---|---|
| A* | 20 | 26.69 | 0.023 |
| ML (DRL) | 23 | 28.43 | 0.011 |
| Increment | 15.00% | 6.51% | ×2.091 |

The same scenario, with the first point changed (as a special flight plan case), is shown in Figure 17. In the horizontal plane, both methods behave similarly, while the main difference appears in the vertical plane due to the altitude change between start and goal. As seen in Table 7, the DRL route is longer (+32%) and slightly less efficient in energy (+12%), but the key result is the computation time: A* requires 35.47 s whereas DRL obtains a solution in only 0.038 s, a reduction of four orders of magnitude. For longer trajectories, A* scales exponentially (see Equation 4), whereas DRL time increases only mildly (see Equation 5), making the difference decisive in this case.




(**a**) (**b**)

**Figure 17.** Evaluation of the 3D model in the first scenario with different altitudes: (**a**) Perspective 1. (**b**) Perspective 2.

 doi:10.20944/preprints202603.0011.v1

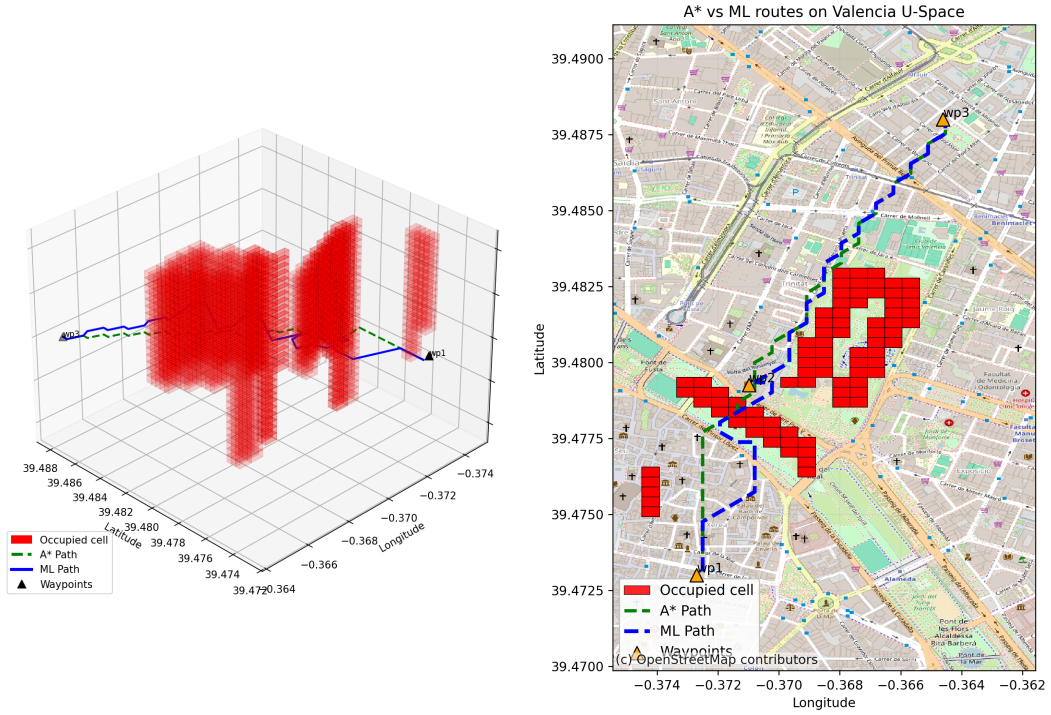


**Table 6.** Numerical results of evaluation in the first scenario (special case of changing altitudes) - 3D.

| Method | Length [steps] | Energy [-] | Time [s] |
|---|---|---|---|
| A* | 66 | 27.93 | 35.47 |
| ML (DRL) | 87 | 31.29 | 0.038 |
| Increment | 32.00% | 12.01% | ×932.42 |

5.2.2. Scenario 2 - 3D

In the next scenario, a flight plan consists of 3 waypoints. Both segments of the route are in conflict with a previously accepted flight. Thus, two different resolutions must be made. In Figure 18, the alternative routes can be seen. In Figure 18b, it is shown that in the first segment, both the DRL and the A* go below the previously accepted flights. In the second segment, both alternative routes avoid the previously accepted flights by going around them.




(**a**) (**b**)

**Figure 18.** Evaluation of the 3D model in the second scenario: (**a**) Perspective 1. (**b**) Perspective 2.

**Table 7.** Numerical results of evaluation in the first scenario (special case of changing altitudes) - 3D.

| Method | Length [steps] | Energy [-] | Time [s] |
|---|---|---|---|
| A* | 47 | 57.67 | 0.056 |
| ML (DRL) | 49 | 65.53 | 0.013 |
| Increment [%] | 4.08 | 13.63 | 76.78 |

## 6. Discussions

The results obtained across both 2D and 3D scenarios confirm that the proposed DRL-based framework constitutes a computationally scalable alternative to classical graph-search methods for strategic conflict resolution in U-space environments.

From a performance perspective, two distinct behaviours can be observed.

In 2D scenarios, the DRL model achieves near-perfect optimality (100% in Scenario 1 and $>$ 99% in Scenarios 2 and 3) while reducing computation time by factors between $\times$2.7 and $\times$4.0. Since the

 doi:10.20944/preprints202603.0011.v1



branching factor in a 2D environment is limited ($b = 8$), the exponential complexity of A* does not yet become as prohibitive as in 3D, at least for the approval stage. However, as mentioned in the *Introduction* (Section 1), in the activation stage, in emergency missions or in DAR situations, the time in which the route is solved must be near-instantaneous, so there, the model is much better and worthy than other algorithms.

In 3D scenarios, a more notable reduction is achieved. In a very specific scenario, a reduction of more than $\times 900$ is achieved. Although such extreme reductions are not representative of the average case, they emerge in specific configurations where significant altitude transitions and volumetric detours are required. In these situations, the search depth $d$ increases substantially and the branching factor in 3D ($b = 26$) amplifies the combinatorial expansion of A*. These scenarios are not the most common in routine cruise-level deconfliction, but they can definitely occur in special cases. Therefore, while the $\times 900$ reduction should not be interpreted as a baseline improvement, it illustrates a structural vulnerability of exponential graph-search methods when confronted with vertically complex resolutions. In contrast, the DRL inference time remains stable and predictable, independently of the number of vertical manoeuvres required.

Beyond the pure computational comparison, the relevance of these results must be interpreted in the operational context of U-space services. In the approval stage, where flight plans are typically submitted hours or even days in advance, the marginal reduction from, for instance, 0.25 s to 0.06 s may not significantly impact system-level performance. In such cases, A* remains a robust and fully deterministic alternative.

However, the activation stage introduces a fundamentally different operational constraint. According to ASTM F3548-21 [14] and the EASA Easy Access Rules for U-space [6], activation responses must be delivered within a strict temporal threshold in the vast majority of cases. In high-density environments, even moderate route length cad lead to exponential growth in A* computation time due to its $\mathcal{O}(b^d)$ complexity. Although this effect is less dramatic in 2D ($b = 8$) than in 3D ($b = 26$), the branching factor still implies rapid scalability limitations as grid resolution increases or as congestion grows.

Therefore, the advantage of the DRL model in 2D scenario should not be interpreted merely as a speed improvement, but rather as a guarantee of bounded inference time. Once trained, the policy execution scales linearly with route length ($\mathcal{O}(d)$), independently of the obstacle configuration. This property is particularly variable in emergency missions or Dynamic Airspace Reconfiguration (DAR) events, where multiple flights may need to be re-evaluated simultaneously and near-instantaneously. In such cases, deterministic but exponentially scaling algorithms may introduce cascading delays that compromise airspace efficiency.

Furthermore, the 2D results demonstrate that a carefully designed reward function, even without explicit energy penalisation, is sufficient to induce near-optimal path behaviour. This confirms that excessive reward complexity is not necessarily beneficial and aligns with findings in the literature that hyperparameter tuning and reward shaping are often more decisive than architectural sophistication. The fact that the 2D model achieves $> 99\%$ optimality without energy terms suggests that the distance-based shaping already embeds an implicit regularisation effect, discouraging unnecessary manoeuvres.

Nevertheless, some limitations remain. Although the DRL consistently reaches the goal in the vast majority of cases, it does not yet provide deterministic guarantees of optimality or completeness. In safety-critical certified systems, this aspect must be carefully addressed, potentially through hybrid architectures where DRL provides rapid candidate solutions and classical algorithms verify feasibility when time allows.

In summary, the 2D experiments confirm that while A* remains suitable for low-density or non-time-critical approval processes, the DRL approach provides a structurally more salable solution for activation-stage conflict resolution, emergency rerouting and dynamically reconfigurable U-space environments. The advantage is not merely numerical but architectural: bounded response time becomes the key operational metric as traffic density increases.

 doi:10.20944/preprints202603.0011.v1



## 7. Conclusions

This paper presented a three-stage framework for strategic conflict resolution in U-space environments, integrating Octree-based airspace discretisation, A* pathfinding and Deep Reinforcement Learning (DRL) models. The approach was designed to address the computational limitations of classical graph-search methods in increasingly dense and volumetric U-space scenarios.

The proposed DRL models were trained on realistic PX4 Software-In-The-Loop trajectories and evaluated on previously unseen scenarios, thereby ensuring generalisation. In 2D environments, the DRL planner achieved near-optimal ($99 - 100\%$ optimality) route lengths while consistently reducing computation time compared to A*. In 3D configurations, where the branching factor and route depth significantly increase computational complexity, the DRL model maintained bounded inference times and avoided the exponential scaling behaviour inherent to classical search algorithms. In specific, vertically complex cases, substantial reductions in computation time were observed.

These results indicate that learning-based planners can provide a scalable alternative for activation-stage strategic deconfliction, particularly in scenarios that require near-instantaneous responses, such as emergency missions or Dynamic Airspace Reconfiguration (DAR). While A* remains a deterministic benchmark for optimality, the DRL approach offers predictable latency and operational feasibility as traffic density and volumetric complexity increase.

Overall, the findings support the integration of carefully designed DRL-based planning modules into future U-space service architectures, thereby contributing to the development of responsive and computationally efficient strategic conflict-resolution mechanisms.

## 8. Future Work

Although the proposed framework demonstrates promising scalability and computational advantages, several extensions are needed to improve its operational maturity and regulatory alignment.

First, the current energy model is intentionally simplified and based on first-order approximations for multirotor platforms. Future work should incorporate higher-fidelity energy-estimation models derived from real flight data, including wind disturbances and vehicle-specific performance envelopes. This would allow the reward formulation to better reflect operational constraints and platform diversity.

Second, the present study focuses on single-flight strategic resolution. In dense U-space environments, multiple simultaneous replanning events may occur, particularly during Dynamic Airspace Reconfiguration (DAR) or emergency prioritisation. Extending the framework to support multi-agent or cooperative resolution strategies is a necessary step for large-scale deployment. This may involve decentralised learning architectures or hybrid coordination mechanisms between deterministic and learning-based planners.

Third, although the DRL model demonstrates strong generalisation in unseen city-scale scenarios, further validation in larger-scale, higher-density simulations is required. Heterogeneous traffic patterns and layered altitude structures should be considered to stress-test scalability and robustness.

Finally, from a certification and safety perspective, future research should explore hybrid verification schemes in which learning-based planners generate candidate routes that can be validated or constrained by deterministic safety filters. Such architectures could facilitate regulatory acceptance while preserving the computational benefits observed in this work.

These directions aim to progressively bridge the gap between experimental validation and operational integration of learning-based strategic conflict resolution services within future U-space ecosystems.

**Author Contributions:** Conceptualization, M.G.; methodology, M.G.; software, M.G., S.A.; validation, M.G., S.A. and A.S.; formal analysis, J.B., S.A., M.G.; investigation, M.G.; resources, M.G., S.A. and A.S.; data curation, M.G. and S.A.; writing—original draft preparation, M.G.; writing—review and editing, J.B. and S.A.; visualization, M.G.; supervision, J.B. and S.A.; project administration, J.B.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

 doi:10.20944/preprints202603.0011.v1



**Funding:** This research was conducted within the framework of U-AGREE project. This project has received funding from the SESAR 3 Joint Undertaking (JU) under grant agreement No 101167187. The JU receives support from the European Union's Horizon Europe research and innovation programme and the SESAR 3 JU members other than the Union.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data available on https://doi.org/10.5281/zenodo.18790882. For further information, contact the corresponding author.

**Acknowledgments:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AI | Artificial Intelligence |
| BADA | Base of Aircraft Data |
| D3QN | Dueling Double Deep Q-Network |
| DAR | Dynamic Airspace Reconfiguration |
| DNN | Deep Neural Networks |
| DQN | Deep Q-Network |
| DRL | Deep Reinforcement Learning |
| EASA | European Union Aviation Safety Agency |
| ENAC | Ente Nazionale per l'Aviazione Civile |
| FAS | Flight Authorisation Service |
| LLA | Latitude, Longitude, Altitude |
| MAC | Mid-Air Collisions |
| MDP | Markov Decision Process |
| ML | Machine Learning |
| NN | Neural Networks |
| PPO | Proximal Policy Optimization |
| RL | Reinforcement Learning |
| SAC | Soft Actor-Critic |
| SFOS | Search for Overlap Service |
| SITL | Software-In-The-Loop |
| UAM | Urban Air Mobility |
| UAS | Unmanned Aircraft Systems |
| USS | UAS Service Supplier |
| USSP | U-space Service Provider |
| UTM | UAS Traffic Management |

## References

1. Imperial War Museums. A brief history of drones.
2. Wagner, W. *Lightning bugs, and other reconnaissance drones*; Aero Publishers, Inc.: Fallbrook, CA, USA, 1982.
3. International Civil Aviation Organization. Annex II – rules of the air. manual, ICAO, Montreal, QC, Canada, 2022. Edition: 11th.
4. International Civil Aviation Organization. Unmanned aircraft systems traffic management (UTM) – a common framework with core principles for global harmonization. manual, ICAO, 2023. Edition: 4th.
5. Commission Implementing Regulation (EU) 2021/664 of 22 April 2021 on a regulatory framework for the U-space (Text with EEA relevance), 2021. Legislative Body: COM, MOVE.
6. Easy Access Rules for U-space - May 2024 | EASA, 2024.
7. F3548 Standard Specification for UAS Traffic Management (UTM) UAS Service Supplier (USS) Interoperability.

 doi:10.20944/preprints202603.0011.v1




8. Obaid, L.; Hamad, K.; Al-Ruzouq, R.; Dabous, S.A.; Ismail, K.; Alotaibi, E. State-of-the-art review of unmanned aerial vehicles (UAVs) and artificial intelligence (AI) for traffic and safety analyses: Recent progress, applications, challenges, and opportunities. *Transportation Research Interdisciplinary Perspectives* **2025**, *33*, 101591. https://doi.org/10.1016/j.trip.2025.101591.
9. Amarillo, S.; Sanchis, A.; Freire, B.F.; Balbastre, J.V. Proposal of UAS Strategic Conflict Detection Concept with a Centralised Service in Multi-USSP Environment Using an Octree Data Structure. In Proceedings of the 2025 Integrated Communications, Navigation and Surveillance Conference (ICNS), 2025, pp. 1–8. ISSN: 2155-4951, https://doi.org/10.1109/ICNS65417.2025.10976940.
10. Sanchis, A.; Garcia, A.; Esparza, S.; Balbastre, J.V. Realistic Trajectory Generation in Simulated Environments for U-Space Systems Assessment. In Proceedings of the 2025 Integrated Communications, Navigation and Surveillance Conference (ICNS), 2025, pp. 1–9. ISSN: 2155-4951, https://doi.org/10.1109/ICNS65417.2025.10976773.
11. International Civil Aviation Organization. Unmanned aircraft systems traffic management (UTM) – a common framework with core principles for global harmonization. Guidance Document, International Civil Aviation Organization (ICAO), 2023. Edition: 4.
12. Cheriet, H.; Badra, K.K.; Samira, C. Comparative Analysis of UAV Path Planning Algorithms for Efficient Navigation in Urban 3D Environments, 2025. arXiv:2508.16515 [cs], https://doi.org/10.48550/arXiv.2508.16515.
13. Li, Q.; Wan, L.; Xiao, B.; Zhou, Y. Research on Multi-Airspace Conflict Detection and Resolution Method Based on Octree Spatial Grid Partitioning. *Aerospace* **2025**, *13*. Company: Multidisciplinary Digital Publishing Institute Distributor: Multidisciplinary Digital Publishing Institute Institution: Multidisciplinary Digital Publishing Institute Label: Multidisciplinary Digital Publishing Institute Publisher: publisher, https://doi.org/10.3390/aerospace13010015.
14. Standard Specification for UAS Traffic Management (UTM) UAS Service Supplier (USS) Interoperability.
15. Gordo, V.; Perez-Castan, J.A.; Sanz, L.P.; Serrano-Mira, L.; Xu, Y. Feasibility of Conflict Prediction of Drone Trajectories by Means of Machine Learning Techniques. *Aerospace* **2024**, *11*. Company: Multidisciplinary Digital Publishing Institute Distributor: Multidisciplinary Digital Publishing Institute Institution: Multidisciplinary Digital Publishing Institute Label: Multidisciplinary Digital Publishing Institute Publisher: publisher, https://doi.org/10.3390/aerospace11121044.
16. He, Y.; Hou, T.; Wang, M. A new method for unmanned aerial vehicle path planning in complex environments. *Scientific Reports* **2024**, *14*, 9257. Publisher: Nature Publishing Group, https://doi.org/10.1038/s41598-024-60051-4.
17. Innan, N.; Kashif, M.; Marchisio, A.; Gan, Y.S.; Barbaresco, F.; Shafique, M. QUAV: Quantum-Assisted Path Planning and Optimization for UAV Navigation with Obstacle Avoidance, 2025. arXiv:2508.21361 [quant-ph], https://doi.org/10.48550/arXiv.2508.21361.
18. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning **2013**.
19. Wang, Z.; Ng, S.X.; EI-Hajjar, M. Deep Reinforcement Learning Assisted UAV Path Planning Relying on Cumulative Reward Mode and Region Segmentation. *IEEE Open Journal of Vehicular Technology* **2024**, *5*, 737–751. https://doi.org/10.1109/OJVT.2024.3402129.
20. Westheider, J.; Rückin, J.; Popović, M. Multi-UAV Adaptive Path Planning Using Deep Reinforcement Learning, 2023. arXiv:2303.01150 [cs], https://doi.org/10.48550/arXiv.2303.01150.
21. Sharma, G.; Jain, S. Deep Reinforcement Learning-Based Framework for Path Planning of AUAVs. *Procedia Computer Science* **2025**, *258*, 1112–1122. https://doi.org/10.1016/j.procs.2025.04.346.
22. Jarray, R.; Bouallègue, S. Improved deep Q-network-based UAV path planning in partially observable environments. *Cluster Computing* **2025**, *28*, 1046. https://doi.org/10.1007/s10586-025-05718-x.
23. Dierkes, J.; Cramer, E.; Hoos, H.H.; Trimpe, S. Combining Automated Optimisation of Hyperparameters and Reward Shape, 2024. arXiv: 2406.18293 [cs] Number: arXiv:2406.18293, https://doi.org/10.48550/arXiv.2406.18293.
24. Mendez, A.P.; Whidborne, J.F.; Chen, L. Wind preview-based model predictive control of multi-rotor uavs using LiDAR. *Sensors* **2023**, *23*. Number: 3711 tex.pubmedid: 37050770, https://doi.org/10.3390/s23073711.
25. European Commission. Commission Implementing Regulation (EU) 2019/947 of 24 May 2019 on the rules and procedures for the operation of unmanned aircraft systems, 2019. tex.howpublished: Official Journal of the European Union.

 doi:10.20944/preprints202603.0011.v1




26. Gong, H.; Huang, B.; Jia, B.; Dai, H. Modelling Power Consumptions for Multi-rotor UAVs, 2022. arXiv:2209.04128 [cs], https://doi.org/10.48550/arXiv.2209.04128.
27. Jacewicz, M.; Żugaj, M.; Głębocki, R.; Bibik, P. Quadrotor model for energy consumption analysis. *Energies* **2022**, *15*. Number: 7136, https://doi.org/10.3390/en15197136.
28. Report Dichiarazioni/Autorizzazioni Operatori APR.
29. Russell, S.J.; Norvig, P. *Artificial intelligence: a modern approach*, fourth edition, global edition ed.; Prentice Hall series in artificial intelligence, Pearson: Boston, 2022.
30. Sutton, R.S.; Barto, A.G. Reinforcement Learning: An Introduction.
31. Liang, S.; Srikant, R. Why Deep Neural Networks for Function Approximation?, 2017. arXiv:1610.04161 [cs], https://doi.org/10.48550/arXiv.1610.04161.
32. Vincent, F.L.; Peter, H.; Riashat, I.; Marc, G.B.; Joelle, P. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning* **2018**, *11*, 219–354. https://doi.org/10.1561/2200000071.
33. Aliferis, C.; Simon, G. Overfitting, Underfitting and General Model Overconfidence and Under-Performance Pitfalls and Best Practices in Machine Learning and AI. In *Artificial Intelligence and Machine Learning in Health Care and Medical Sciences: Best Practices and Pitfalls*; Simon, G.J.; Aliferis, C., Eds.; Springer International Publishing: Cham, 2024; pp. 477–524. https://doi.org/10.1007/978-3-031-39355-6_10.
34. Statnikov, A.; Aliferis, C.; Hardin, D.; Guyon, I. *A Gentle Introduction to Support Vector Machines in Biomedicine: Volume 1: Theory and Methods*; 2011. Pages: 183, https://doi.org/10.1142/7922.
35. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. Publisher: Nature Publishing Group, https://doi.org/10.1038/nature24270.
36. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning*; MIT Press, 2016.