Green Energy and Intelligent Transportation 3 (2024) 100157

Contents lists available at ScienceDirect

Green Energy and Intelligent Transportation

journal homepage: www.journals.elsevier.com/green-energy-and-intelligent-transportation

Full length article

# A reinforcement learning approach to vehicle coordination for structured advanced air mobility

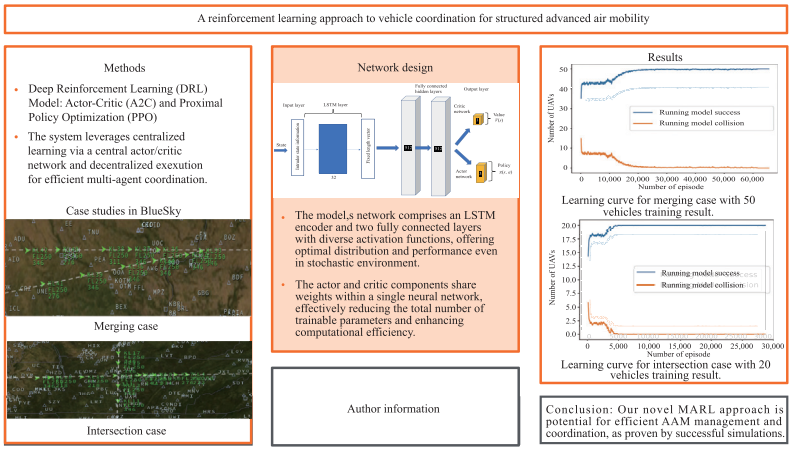
Sabrullah Deniz, Yufei Wu, Yang Shi, Zhenbo Wang *

*Department of Mechanical, Aerospace, and Biomedical Engineering, The University of Tennessee, Knoxville, TN, 37996, USA*

## HIGHLIGHTS

- A novel deep reinforcement learning approach to safe and efficient AAM traffic separation.
- A new MARL framework for AAM vehicle coordination in merging and intersection scenarios.
- Trade-off studies that reveal impacts of network design and hyperparameters on the performance of the algorithms.
- Extensive simulations that demonstrate the performance of the proposed methods.

## GRAPHICAL ABSTRACT




## ARTICLE INFO

*Keywords:*
Advanced Air Mobility (AAM)
Urban Air Mobility (UAM)
Air Traffic Control (ATC)
Multi-Agent Reinforcement Learning (MARL)

## ABSTRACT

Advanced Air Mobility (AAM) has emerged as a pioneering concept designed to optimize the efficacy and ecological sustainability of air transportation. Its core objective is to provide highly automated air transportation services for passengers or cargo, operating at low altitudes within urban, suburban, and rural regions. AAM seeks to enhance the efficiency and environmental viability of the aviation sector by revolutionizing the way air travel is conducted. In a complex aviation environment, traffic management and control are essential technologies for safe and effective AAM operations. One of the most difficult obstacles in the envisioned AAM systems is vehicle coordination at merging points and intersections. The escalating demand for air mobility services, particularly within urban areas, poses significant complexities to the execution of such missions. In this study, we propose a novel multi-agent reinforcement learning (MARL) approach to efficiently manage high-density AAM operations in structured airspace. Our approach provides effective guidance to AAM vehicles, ensuring conflict avoidance, mitigating traffic congestion, reducing travel time, and maintaining safe separation. Specifically, intelligent learning-based algorithms are developed to provide speed guidance for each AAM vehicle, ensuring secure merging into air corridors and safe passage through intersections. To validate the effectiveness of our proposed model, we conduct training and evaluation using BlueSky, an open-source air traffic control simulation environment. Through the simulation of thousands of aircraft and the integration of real-world data, our study

* Corresponding author.
*E-mail address:* zwang124@utk.edu (Z. Wang).

https://doi.org/10.1016/j.geits.2024.100157
Received 19 July 2023; Received in revised form 9 November 2023; Accepted 10 November 2023
Available online 11 January 2024

demonstrates the promising potential of MARL in enabling safe and efficient AAM operations. The simulation results validate the efficacy of our approach and its ability to achieve the desired outcomes.

## 1. Introduction

### 1.1. Background and motivation

The global population growth has led to an increase in the need for mobility, resulting in surface traffic congestion. Traffic congestion negatively affects the environment, mobility, accessibility, and socio-economic activity. Infrastructure expansions and adaptive traffic control have been primarily used to reduce road congestion. In response to ground traffic congestion, Advanced Air Mobility (AAM) and Urban Air Mobility (UAM) have gained popularity in recent years due to several factors, including technological advances, sustainability, traffic congestion, increased demand for air transportation, and investment. AAM is a novel concept that leverages new aircraft designs and revolutionary flight technologies to transport people and cargo between places in unexplored or underexplored airspace. UAM, a subset of AAM, focuses on air transportation services in urban areas. Both aim to improve efficiency, sustainability, and reduce environmental impact [1].

Leading government agencies and corporate entities, including NASA, Uber, Airbus, Volocopter, Bell, and Embraer, are developing their AAM/UAM concepts as AAM and UAM become more promising for future transportation [2–5]. Electric vertical takeoff and landing (eVTOL) vehicles that utilize three-dimensional airspace for personal commutes or on-demand air taxis are the most popular concept. An eVTOL aircraft can take off without using a runway. It is considerably more energy efficient than traditional helicopters and can carry three, five, or even more passengers depending on the manufacturer and vehicle design. The eVTOL vehicles are powered by electricity, which minimizes emissions and enables an energy-efficient solution to alleviate surface traffic congestion.

The initial AAM ecosystem will make use of the pre-existing helicopter infrastructure, encompassing established routes, helipads, and Air Traffic Control (ATC) services. Millions of unmanned aerial vehicles (UAVs) are expected to operate in U.S. airspace by 2040, according to a report by the Federal Aviation Administration (FAA) [6]. As a result, the workload of ATC is constantly increasing. The traffic management of AAM vehicles is a crucial issue that must be resolved in order to maintain airspace safety and alleviate airspace traffic congestion. The existing ATC system, which relies on human controllers to handle air traffic, presents significant challenges and complexities. The requirement for human operators to multitask in managing the immense volume of air traffic renders the task laborious and formidable. In 2018, the FAA NextGen Office published a preliminary concept of operations (ConOps) for UAS Traffic Management (UTM), facilitating low-altitude UAS operations. The FAA and NASA have updated the UTM ConOps to v2.0 [7], aiming to increase airspace capacity for more intricate UTM operations.

The first published ConOps v1.0, the FAA and NASA outlined that UAM operations would occur within an air corridor established and announced by the FAA. However, it was emphasized that the industry and other stakeholders would play a key role in modifying this corridor. AAM corridors are envisioned as effective and secure channels to facilitate high-density air traffic operations, ensuring both safety and efficiency. The vehicles entering the AAM corridors are required to comply with stringent performance standards and regulations, which may vary based on the specific characteristics of the local airspace and vary from one corridor to another [8]. The corridors will either merge or intersect alongside the route. ATC will evaluate the availability of corridors based on adjacent conditions and activities; however, no tactical separation facilities will be provided within corridors, and pilots in command will be responsible for ensuring safe operations [8]. Despite substantial development, many technical issues remain unresolved [9,10]. In this research, we aim to address one of the key research challenges, i.e., *how to safely and efficiently manage the movements of multiple AAM vehicles at a merging point and an intersection within a large-scale structured airspace consisting of multiple airways or air corridors* [8]. At a merging point, vehicles operating on a secondary air corridor merge into the main corridor, as shown in Fig. 1. At a two-dimensional (2-D) AAM intersection, vehicles approach the intersection from different directions and pass the intersection as shown in Fig. 2. The primary research focus of our study refers to the management of traffic within the corridors, specifically at the merging points and intersections.

In this paper, it is assumed that vehicle operations are handled by a highly trained reinforcement learning agent, enabling fully autonomous piloting and flight. Reinforcement learning has attracted considerable interest, primarily driven by the impressive performance demonstrated by reinforcement learning agents such as AlphaGo [11] and OpenAI [12]. AlphaGo made history as the first computer program to defeat skilled human Go players, including a world champion. This notable progress in AI has shown the underlying theoretical foundations and computational capacities of intelligent agents and AI technologies, which have the potential to enhance and facilitate human tasks. Reinforcement learning, as a model-free self-learning algorithm, presents significant potential for advanced air traffic control and management. By enabling an agent (such as an AAM vehicle) to learn the optimal policy through interactions with its environment (i.e., airspace) and maximizing cumulative rewards, reinforcement learning offers a promising approach. The reward serves as feedback from the environment, providing the agent with positive reinforcement for good behavior and imposing penalties for undesirable behavior. In the domain of AAM management and control, it is feasible to generate real-time conflict resolution advisories for aircraft by framing the tasks typically performed by human air traffic controllers as a

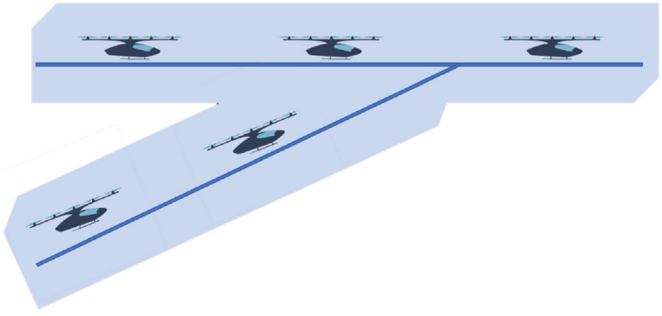


**Fig. 1.** AAM merging scenario.

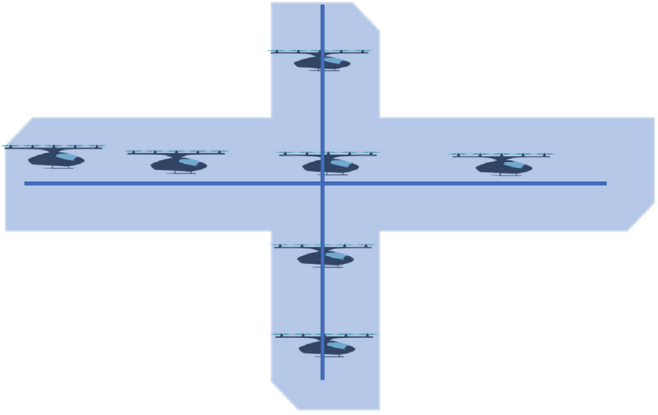


**Fig. 2.** AAM intersection scenario.

reinforcement learning problem that can be solved with minimal computation time. Moreover, to apply reinforcement learning in organizing AAM traffic flow, which involves tasks such as routing and collision avoidance, it becomes crucial to have a platform capable of simulating air traffic management. This platform allows for the testing and evaluation of the effectiveness of reinforcement learning models in this context.

This work presents a deep multi-agent reinforcement learning (MARL) framework that enables autonomous AAM traffic separation to minimize conflicts during flight. The framework employs a centralized learning approach and decentralized execution for efficient decision-making by AAM agents. The proposed framework has the capability to effectively manage a variety of vehicles at merging points or intersections by offering speed advisories to each vehicle within the air corridor. To store and retain information about the environment, a Long Short-Term Memory (LSTM) network [13] is employed. This LSTM network captures and encodes all pertinent information into a fixed-length vector, facilitating comprehensive representation of the environment. This work has the potential to effectively address the autonomous traffic control problem for AAM vehicles, facilitating smooth merging and conflict-free passage at intersections within the air corridors. Furthermore, the performance of the developed model in this paper is demonstrated using the BlueSky air traffic control simulator [14]. This simulator enables the vehicle to interact with the environment, providing a platform to showcase the model's effectiveness.

### 1.2. Related work

During the flight, ATC systems must ensure that a safe distance between the vehicles must be maintained. Conflict is deemed to exist when horizontal and vertical distances are simultaneously smaller than the minimum separation. There are usually three kinds of conflict resolution methods to handle safe separation problems between aircraft by adjusting heading, altitude, and speed. Heading adjustment can be used for conflict resolution by changing the heading angle, which turns the aircraft right or left in the angle. This method will change the aircraft route and may lead to a delay in arriving the destination when there are many aircraft in the conflict zone. Adjusting altitude is one of the most promising methods for separation problems. The conflicts will be determined in advance, and the aircraft will make necessary adjustments to its flight level to avoid them. Lastly, speed adjustment for the aircraft at the same flight level will solve most of the conflict resolution problems by only adjusting the speed around the aircraft's cruise speed. In this paper, we use the speed adjustment method for safe separation of AAM vehicles in the conflict zone with the help of deep reinforcement learning approaches.

In addition to traditional rule-based and optimization-based approaches, deep reinforcement learning has been successfully applied to merging control on highways and traffic control at signalized intersections in ground transportation. These applications aim to minimize travel delays for travelers and enable safe autonomous passage at merging roadways and intersections [15–17]. For ground vehicles, entering into congested traffic lanes is a significant difficulty. To ensure a secure merging process, one strategy is to position an agent at the merging point, which would provide speed advisories to the vehicles. Alternatively, each vehicle could operate as an independent agent within the environment, making its own decisions regarding merging time and speed [18]. Our research provides speed advisories to each AAM vehicle to prevent conflicts at merging points and intersections. Each AAM vehicle acts as an agent, and a learning-based algorithm is employed to offer speed advisories to effectively manage potential conflicts.

Autonomous ATC has been researched and implemented for AAM traffic control. An auto-resolver was developed in previous works [19,20] to iteratively compute air traffic trajectories. This auto-resolver tests different candidate trajectories until a suitable trajectory is identified that satisfies all conflict resolution conditions. This detecting system also contains aircraft speed, altitudes, flight plans, and locations. Through the utilization of a physics-based auto-resolver approach, the detection and resolution of conflicts were achieved with commendable performance. In another work, the researchers employed a multi-agent method to solve intersection problems for AAM by using reinforcement learning with a comprehensive reward function [21]. In that approach, the authors treated each agent independently in a 2-D space, addressing computational limitations by considering fixed locations. They utilized three distinct actions for airspace navigation: adjusting aircraft separation, managing departures and ground delay, and rerouting aircraft. By applying reinforcement learning, agents learned optimal actions, resulting in congestion reduction of up to 80%. However, limitations in the learning process suggested the possibility of developing more efficient and effective methodologies.

Advancements in deep learning techniques and computer hardware have facilitated the computation and evaluation of real-time multi-agent policies in increasingly realistic environments. Air Traffic Control (ATC) automation using Multi-Agent Reinforcement Learning (MARL) was formulated recently in Ref. [22] introduced the concept of utilizing MARL for ATC automation. In that work, a deep MARL framework was proposed to handle the separation problem for ATC. The specific focus was on solving the air traffic intersection problem by formulating it as an MARL model and leveraging the DD-MARL framework. This approach demonstrated remarkable performance in addressing complex sequential decision-making problems characterized by uncertainty, highlighting its potential for enhancing ATC operations. Their results showed the great potential of the MARL model for high-performance ATC. Furthermore, current research in the area of ground transportation for merging control applies comparable methodologies to identify the most optimal merging strategies. In Ref. [18], the ground transportation merging problem was addressed through the formulation of a multi-agent model. This problem was approached as a model-free scenario and resolved using a multi-agent network as described in Ref. [23]. In this decentralized MARL model, each agent's observation is limited to a specific portion of the environment, allowing communication only with nearby neighbors. However, this restricted observation capability may lead to traffic flow instability and might not be suitable for scenarios involving heavy traffic merging.

In the field of UAM and ATM, a comprehensive retrospective review was conducted in Ref. [24] research covering work from 1995 to 2022. The review highlighted the evolution of airspace users, starting from helicopters to UAVs and passenger-carrying UAVs for UAM. It emphasized the complexity of integrating various vehicle types and high-density operations in urban environments. The review outlined the multidisciplinary nature of implementing UAM systems, addressing vehicle design, certification, airspace integration, operational aspects, infrastructure requirements, and public acceptance. It also stressed the importance of advanced U-space services, strategic conflict avoidance, and efficient airspace management in the context of UAM. A comprehensive survey on deep learning in Air Traffic Management (ATM) in Ref. [25] provides valuable insights into various deep learning applications in ATM, opportunities, and open challenges. This survey offers a detailed examination of how deep learning techniques are applied within ATM, including their potential applications and areas that require further research. It serves as a valuable resource for understanding the landscape of deep learning in ATM and can be used as a reference for comparing different methods and approaches in the field. In another research [26], the authors address the challenges of airspace deconfliction in the context of AAM and explores the need for strategic and tactical coordination to ensure safe and efficient operations. It highlights the use of predefined airways or corridors in conventional ATM and their limitations when applied to the dense and complex operations of AAM. The authors propose an alternative approach by examining the feasibility of not using corridors, backed by a simple and scalable simulation model. The paper also discusses the importance of redundancy and diversity in traffic coordination methods for enhanced system safety. It identifies a

gap between cooperative ground-based approaches and non-cooperative, aircraft-centered methods like Detect-And-Avoid (DAA). The paper introduces an airborne cooperative method that aims to safely resolve conflicts involving multiple aircraft, offering improved efficiency compared to DAA alone. This method could serve as an alternative or supplementary approach to ground-based traffic coordination.

Despite the significant advances in deep reinforcement learning (DRL), its application to ATM systems is facing several formidable challenges. It is crucial to acknowledge these challenges to provide context for the research presented in this paper. While our proposed method aims to address specific aspects of these challenges, it does not resolve all of them comprehensively. Below, we outline the primary challenges in applying DRL to ATM: 1) High Dimensionality: the air traffic management system is a highly complex, dynamic environment with many state and action variables, making it challenging for DRL algorithms to learn optimal policies in a reasonable amount of time; 2) Data Requirements: unlike traditional machine learning approaches, DRL methods, including the one proposed in this paper, do not rely on extensive pre-existing datasets for training. Instead, they learn from interactions with the environment. However, data collection in ATM can present its own challenges, and the quality and quantity of data can impact the effectiveness of DRL methods; 3) Safety: ensuring the safety of passengers and crew is paramount in ATM. This challenge involves not only training DRL algorithms to optimize policies but also guaranteeing the safety of the learned policies, which can be a complex and critical concern; 4) Real-time Constraints: air traffic management is a real-time system with strict time constraints, and the DRL algorithms must operate in real-time, making the learning process and policy optimization challenging; 5) Complex Interactions: the interactions between aircraft and between aircraft and ATC are complex, and it is challenging for DRL algorithms to capture these interactions accurately; 6) Regulation: the aviation industry is highly regulated, and any new technologies must comply with existing regulations and standards. The integration of DRL algorithms into air traffic management systems must comply with these regulations. These challenges must be addressed to ensure the successful implementation of DRL algorithms in air traffic management systems and realize DRL's potential benefits for AAM.

Motivated by the previous work in Ref. [22] and building on our preliminary results in Refs. [27,28], we develop a novel MARL model in this paper for coordinating AAM vehicles under two scenarios – merging points and intersections – with an aim of enabling a more realistic AAM control model. Our model, as mentioned earlier, will operate in structured airways or air corridors within the AAM concept proposed by NASA and Uber Elevate [29]. The main contribution of this work is threefold. First, a novel deep MARL framework is designed to enable safe and efficient AAM traffic separation and reduce conflicts during flight based on a centralized learning, decentralized execution approach. Second, the proposed MARL framework is applied to mitigate the potential risks and optimize the movements of AAM vehicles in real time under the merging and intersection scenarios by allowing the vehicles to interact with the dynamic AAM environment and the changing traffic conditions. Third, the effect of the network design and hyperparameters on the performance of the algorithms is investigated. Furthermore, the proposed framework's efficacy and performance in generating real-time conflict resolution advisories are demonstrated through its implementation in the BlueSky air traffic control simulator.

The rest of this paper is structured as follows. Section 2 provides an introduction to the foundational concepts, including reinforcement learning, policy-based learning, and MARL. System models and problem formulation are detailed in Section 3. In Section 4, the network design and parameter tuning of our deep MARL framework are presented in great details. Section 5 introduces the simulation environment and shows the outcomes that effectively illustrate the efficacy of our proposed model. Finally, in Section 6, this paper concludes by summarizing the key findings and contributions discussed throughout the study.

## 2. Preliminaries

### 2.1. Reinforcement learning

Reinforcement learning is a distinct type of machine learning that differs from both supervised learning and unsupervised learning approaches [30]. Reinforcement learning uses training data to evaluate the actions taken, while other types of learning use the data to correct actions as a reference. Reinforcement learning problems can be treated as Markov Decision Process (MDP) [31]. There has been a growing interest in the application of reinforcement learning techniques to address challenges related to vehicle coordination and management in both the fields of ground transportation and air mobility. In reinforcement learning, an agent interacts with unknown environments, learning policies through trial and error by receiving rewards based on its actions. The objective is to maximize the cumulative rewards in an interactive environment, where the agent's actions influence the rewards it receives. The reinforcement learning model can be represented using the variables states $S$, actions $A$, reward function $R$, and discount factor $\gamma$. In reinforcement learning, the learning component involves finding the optimal actions for each state to maximize rewards. The ultimate objective is to discover an optimal policy, denoted as $\pi^*$, which maximizes the cumulative rewards for future steps starting from any initial state:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{T}(r(s_t, a_t))\middle|\pi\right] \tag{1}$$

where $t$ represents the current time, $T$ represents the total time, and $\pi$ represents the policy. By formulating the reward function and maximizing the cumulative rewards, the optimal solution can be reached.

### 2.2. Policy-based learning

There are two fundamental algorithms in reinforcement learning, particularly value-based algorithms and policy-based algorithms. Value-based algorithms, such as Deep Q-Network (DQN) and its variations, and policy-based algorithms, including Deep Deterministic Policy Gradient (DDPG) [32] and Proximal Policy Optimization (PPO) [33], can solve conflict resolution problems. In our study, we employ the actor-critic (A2C) reinforcement learning algorithm [34] based on policy iteration and compare the performance of the algorithm with DDPG. These algorithms learn stochastic policies in a model-free environment, allowing it to effectively handle unknown environments and uncertainties arising from other agents' actions, which is a distinguishing advantage compared to value-based algorithms [35]. In addition, to address learning instability and improve performance, we combine A2C learning with PPO optimization in this paper. Specifically, we use A2C for policy learning and value learning and use the PPO algorithm for policy optimization. The PPO algorithm is a cutting-edge policy optimization technique that was recently developed. It is simple to adjust, performs well, updates at each step, and minimizes the cost function for the problem being solved. PPO optimizes the policy function by limiting the update to a certain trust region, resulting in more stable training and better exploration of the action space. We will take advantage of the benefits of these algorithms to improve the performance and stability of our reinforcement learning tasks by combining the A2C learning technique with PPO optimization.

While A2C and PPO are well-suited for tasks with discrete action spaces and high-dimensional state spaces, there exist many reinforcement learning tasks that require dealing with continuous action spaces. One of the state-of-the-art algorithms tailored for such tasks is the DDPG [36]. DDPG is an actor-critic algorithm that extends the ideas of DPG (Deterministic Policy Gradient) to leverage deep neural networks for approximating both policy and value functions. Since the action space is continuous, the policy outputs a deterministic value for each action given a state, rather than a distribution over actions.

- Policy network (actor): the actor network in DDPG is deterministic. For a given state $s_t$, the actor network $\mu(s_t|\theta_\mu)$ outputs a specific action. This contrasts with the stochastic policy outputs in A2C;
- $Q$-value network (critic): the critic network computes the action-value function $Q(s_t, a_t|\theta_Q)$ which provides an estimate of the expected return for taking action $a_t$ in state $s_t$.

DDPG also introduces the use of target networks for both the actor and critic to stabilize training. The target networks are copies of the actor and critic networks but with slowly updated parameters. The objective is to adjust the policy parameters $\theta^\mu$ to maximize the expected reward:

$$\max_{\theta^\mu} \mathbb{E}_{s_t\sim\rho^\beta, a_t\sim\mu}[Q(s_t, a_t|\theta^Q)] \tag{2}$$

The critic is updated by minimizing the mean squared error between the estimated $Q$-value and the target $Q$-value:

$$L(\theta^Q) = \mathbb{E}_{s_t\sim\rho^\beta, a_t\sim\mu}\left[(Q(s_t, a_t|\theta^Q) - y_t)^2\right] \tag{3}$$

where:

$$y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta_{\mu'})|\theta_{Q'}) \tag{4}$$

$Q'$ and $\mu'$ denote the critic and actor target networks, respectively. The target $Q$-value, $y_t$, is the reward $r_t$ plus the discounted expected return from the next state.

One major advantage of DDPG over traditional methods is its ability to handle tasks with high-dimensional continuous action spaces without the need for action discretization. Moreover, by leveraging experience replay and target networks, DDPG can achieve more stable and faster convergence.

Value-based reinforcement learning algorithms implicitly find the optimal policy by finding the optimal value function, while policy-based reinforcement learning algorithms directly optimize the objective function. Policy-based reinforcement learning is very effective in learning stochastic policies in high-dimensional and stochastic continuous action spaces. The actor-critic method [37] is a combination of policy learning and value learning. While the policy function plays the actor's role, the value function is in the critic role. The actor-critic method has two networks. In the A2C algorithm, the actor-network is responsible for determining which action to take, while the critic-network provides feedback to the actor by evaluating the quality of the chosen action and providing guidance on how to adjust it to achieve the optimal policy.

Actor-critic is similar to REINFORCE algorithm [38], which is one of the policy gradient algorithms. In the REINFORCE algorithm, the policy parameter is updated through Monte Carlo updates, which take random samples. The policy gradient expression for REINFORCE algorithm is:

$$\Delta J(\theta) = \Delta \mathbb{E}_{\pi_\theta}[R(\tau)] = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T-1} \Delta_\theta \log \pi_\theta(a_t|s_t)\right) R(\tau)\right] \tag{5}$$

The advantage function for REINFORCE algorithm is as follows:

$$A(s_t, a_t) = \sum_{t'=0}^{T-1} r_{t'} - b(s_t) \tag{6}$$

where $R(\tau) = \sum_{t=0}^{T-1}$ represents the reward of the trajectory, and $b(s_t)$ represents the baseline function [39].

The A2C algorithm combines both policy-based and value-based reinforcement learning. It learns a stochastic policy that maps states to actions, and a state-value function that estimates the expected return from a given state. The value function is optimized to provide a precise estimate of the expected return, and the policy is optimized to choose behaviors that maximize the expected return.

The expected return is defined as the sum of discounted rewards from the current time step $t$ to the end of the episode $T$, given a state $s_t$ and action $a_t$:

$$R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k} \tag{7}$$

where $\gamma$ is the discount factor that determines the importance of future rewards, and $r_{t+k}$ is the reward received at time step $t + k$.

The actor network in A2C outputs a probability distribution over actions given the current state. It is defined as:

$$\pi_\theta(a_t|s_t) = \text{softmax}\left(\boldsymbol{f_\theta}(\boldsymbol{s_t})^{\mathrm{T}} \boldsymbol{g_\theta}(\boldsymbol{a_t})\right) \tag{8}$$

where $\theta$ represents the weights of the actor network, $\boldsymbol{f_\theta(s_t)}$ is the feature vector of the state, $\boldsymbol{g_\theta(a_t)}$ is the parameter vector of the action, and softmax is a function that normalizes the output to sum to one.

The critic network in A2C estimates the expected return from a given state. It is defined as:

$$V_\theta(s_t) = \boldsymbol{f_\theta}(\boldsymbol{s_t})^{\mathrm{T}} \boldsymbol{h_\theta} \tag{9}$$

where $\boldsymbol{h_\theta}$ is the parameter vector of the critic network.

The advantage function $A_t$ measures how much better the chosen action was compared to the expected value from the current state, and is defined as:

$$A_t = R_t - V_\theta(s_t) \tag{10}$$

The policy and value function parameters are updated using the following formulas:

$$\Delta\theta_{\text{policy}} = \alpha_{\text{policy}} \nabla_\theta \log \pi_\theta(a_t|s_t) A_t \tag{11}$$

$$\Delta\theta_{\text{value}} = \alpha_{\text{value}} \nabla_\theta (V(s_t) - V_\theta(s_t))^2 \tag{12}$$

where $\alpha_{\text{policy}}$ and $\alpha_{\text{value}}$ are the learning rates for the policy and value networks, respectively.

The A2C algorithm's learned policy settings are further optimized by the PPO algorithm. PPO maximizes a surrogate objective function that approximates the ratio of the new policy to the old policy in order to improve the policy. The clipped surrogate objective is defined as:

$$L^{\text{clip}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{13}$$

where $r_t(\theta)$ is the ratio of the new policy to the old policy, and $\epsilon$ is a hyperparameter that determines the size of the trust region.

The policy is then updated by maximizing this objective function using stochastic gradient ascent. The update is constrained by a trust region, which ensures that the updated policy does not deviate too far from the old policy. The trust region is controlled by the hyperparameter $\epsilon$ in the clipped surrogate objective function. By limiting the updates to a certain trust region, PPO is able to achieve more stable training and better exploration of the action space.

To further improve the training of the A2C algorithm, PPO also uses the value function to estimate the advantage function. The advantage function measures how much better the chosen action was compared to the expected value from the current state. It is defined as:

$$A_t = R_t - V_\theta(s_t) \tag{14}$$

where $R_t$ is the total-discounted reward received after taking action $a_t$ in the state $s_t$, and $V_\theta(s_t)$ is the estimate of the state-value function for the current state.

The value function is updated using the standard temporal difference update rule:

$$V_{\theta'}(s_t) = V_\theta(s_t) + \alpha_v(R_t - V_\theta(s_t)) \tag{15}$$

where $V_{\theta'}(s_t)$ is the updated estimate of the state-value function, and $\alpha_v$ is the learning rate for the value network.

Finally, the policy and value function parameters are updated using the calculated gradients. The policy is updated using the clipped surrogate objective function:

$$\theta \leftarrow \theta + \Delta\theta_{\text{policy}} \tag{16}$$

where $\Delta\theta_{\text{policy}}$ is the gradient of the policy function. The value function is updated using the mean squared error loss:

$$\theta \leftarrow \theta + \Delta\theta_{\text{value}} \tag{17}$$

where $\Delta\theta_{\text{value}}$ is the gradient of the value function.

In summary, the A2C algorithm utilizes two neural networks to learn the policy and state-value function, combining policy-based and value-based reinforcement learning. The PPO algorithm is used to further optimize the policy parameters learned by the A2C algorithm by maximizing a clipped surrogate objective function that approximates the ratio of the new policy to the old policy. The advantage function is used to measure how much better the chosen action was compared to the expected value from the current state, and the value function is updated using the standard temporal difference update rule. By combining the benefits of A2C and PPO, we can achieve more stable training, better exploration of the action space, and improved performance in our reinforcement learning tasks.

In the upcoming algorithm performance subsection, we present a comprehensive comparison of the performance of both A2C combined with PPO and DDPG. The motivation behind this is to understand the strengths and weaknesses of each algorithm in the context of our specific reinforcement learning tasks. While A2C with PPO offers robustness in environments with discrete action spaces and uncertainties arising from other agents, DDPG shines in tasks with continuous action domains. The comparative analysis provides insights into the practical efficacy of these algorithms, guiding future endeavors in choosing the appropriate algorithm for similar tasks.

### 2.3. Multi-agent reinforcement learning

Multi-agent reinforcement learning (MARL) has demonstrated remarkable achievements in various multi-agent systems, including traffic light control, games, power grid control, and more [23,40]. MARL is a framework that involves the interaction between multiple agents within a shared environment, as well as their interactions with each other. In contrast, single-agent reinforcement learning focuses solely on the interaction between a single agent and the environment [41]. One of the main challenges in MARL is that each agent has its own individual goal to achieve within the shared environment. This goal might be unknown to other agents, leading to increased complexity in the decision-making process for each agent. Every new agent significantly expands the problem's complexity [42]. One of the well-suited approaches to address MARL problems is independent *Q*-learning. In this method, each agent maintains its own action-observation system and network parameters, treating the other agents as components of the environment [43]. However, the independent *Q*-learning method may face challenges, as changes in the policy of one agent can impact the policies of other agents. This interdependence can lead to instability and difficulties in achieving desired coordination among the agents [44]. To handle this learning instability, a solution involves training a group of agents in a centralized manner with an open communication channel for coordinated learning [45]. To effectively interact and resolve agent negotiations, the communication channel is crucial. In this work, we use the A2C method to address learning instability concerns, while utilizing the Proximal Policy Optimization (PPO) algorithm to approximate the policy and value functions for each agent [33]. Additionally, we conduct a comparative study with the Deep Deterministic Policy Gradient (DDPG) algorithm to evaluate and contrast the performances of both methodologies in our reinforcement learning tasks.

## 3. System model and problem formulation

The system model and problem formulation refer to the mathematical representation and definition of the problem to be solved. In the context of AAM management and deep reinforcement learning (DRL), the system model would describe the components of the air traffic management system and how they interact with each other. The problem formulation would define the objective of the system, such as ensuring safe separation between aircraft, and how that objective can be achieved through the use of DRL. The system model and problem formulation are important because they provide the foundation for developing and evaluating the effectiveness of the DRL approach for AAM. By defining the problem and the components of the system, the DRL algorithm can be designed and tested in a controlled environment to ensure that it can meet the system's objective.

As discussed above, this research focuses on the applicability of DRL for AAM merging and intersection case scenarios for safe separation between AAM vehicles during the flight. The DRL model comprises a deep neural network and an agent interacting with the environment. The relative position and velocity of each aircraft define the state of the system. The actions of the agent are the speed adjustments for each aircraft, which are selected based on the state of the system. The reward function is designed to incentivize safe separation between aircraft and to minimize any potential conflicts. The objective of the reinforcement learning model is to learn a policy that maps the state of the system to the best action to take to maximize the total reward over time. The reinforcement learning model is trained using a trial and error approach, where the agent interacts with the environment, selects actions based on the state of the system, and receives rewards based on the outcomes of its actions. Over time, the agent learns a policy that maps the state of the system to the best action to take to maximize the total reward. This policy is used to control the speed adjustments of each aircraft during the flight, to ensure safe separation and to minimize any potential conflicts.

The BlueSky simulator environment [14] is used for demonstrating the performance of the reinforcement learning model in this research. This simulation software is not built for eVTOL vehicles, but it provides a realistic environment for testing the effectiveness of the proposed deep MARL framework for autonomous air traffic separation. BlueSky allows the agent to interact with the environment, making it an ideal platform for testing the performance of the model developed for safe separation between vehicles during the flight. We develop two case studies to test our MARL model for merging vehicles at a single point and an intersection scenario. There are several different aircraft types in the BlueSky simulator. In the BlueSky simulation, we choose Airbus A318 as the vehicle type for all the aircraft. In our assumptions, it is considered that all aircraft maintain their designated routes throughout the flight. Fig. 3 illustrates the interaction between an agent and the environment, as well as the structure of the reward function that guides decision-making throughout its flight.

### 3.1. Problem formulation

A multi-agent system refers to a collection of autonomous entities that operate within the same environment, with each agent striving to achieve its individual goals within that shared environment [46]. Multi-agent systems have been used in diverse applications, including robotics, data mining, distributed control, resource management, and game planning [46]. In our research, we use multi-agent systems concept to understand the behavior of each AAM vehicle during the flight to avoid potential conflicts. The goal of the MARL model is to learn a policy that maximizes the expected cumulative reward of each agent over a long

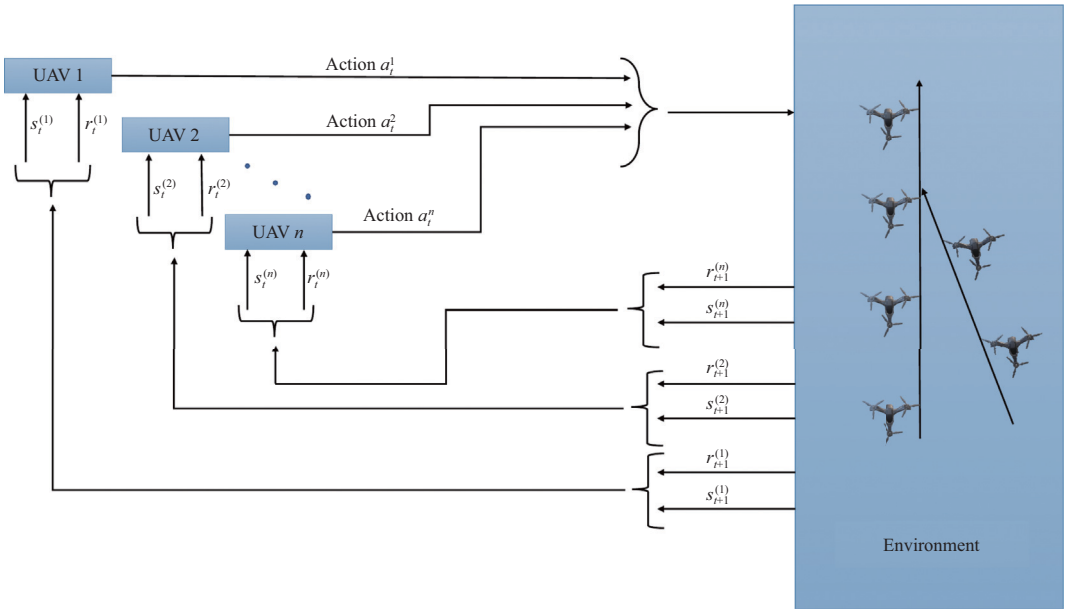



**Fig. 3.** Multi-agent reinforcement learning scheme.

period of time. The policy maps each state to a set of actions that the agent should take to maximize the reward.

In the AAM traffic control problem considered in this paper, the state space and action space are continuous; therefore, the policy is typically represented by a deep neural network. The problem is framed as an MARL problem, where each AAM vehicle is treated as an agent. The model algorithm runs on each vehicle, offering speed guidance to ensure safe and timely arrival at the destination by avoiding the long waiting times and potential conflicts at merging points and intersections. In the upcoming subsections, we will outline the definitions of states, actions, and reward structures for each agent.

### 3.2. State space

The state contains the necessary information for an agent to make decisions. In our context, each AAM vehicle is considered an agent that bases its decision-making on the information stored in the state. We assume that the position and the dynamics of each agent are available continuously. In our scenario, the ownship represents the vehicle approaching the merging point from either the main or merging air corridors. The intruders, on the other hand, are the nearest vehicles in either the main or merging corridor in relation to the ownship. These intruders can be positioned ahead or behind the ownship within the corridors. To capture the intruder information effectively, an LSTM network is used. This network processes intruder details such as speed, acceleration, distance to the goal, distance to the merging point, and distance between intruders, and converts them into a fixed-length vector. Subsequently, the LSTM network encodes this information and is trained to assist the ownship in determining which intruder(s) require consideration.

To formulate the state and intruder state information for the agents, we adopt a similar approach as outlined in Ref. [22]:

$$S_t^0 = (d_{\text{goal}}^0, v^0, a^0, r^0, d^{\text{LOS}}) \tag{18}$$

$$h_t^0(i) = (d_{\text{goal}}^i, v^i, a^i, r^i, d_0^i, d_{\text{int}}^0, d_{\text{int}}^i) \tag{19}$$

where $S_t^0$ represents the state of the ownship, and $h_t^0(i)$ represents the information of the $i$th intruder that is available at time $t$. Other involved variables are defined below.

$d_{\text{goal}}^0$ : Distance to the ownship's goal
$v^0$ : Velocity of the ownship
$a^0$ : Acceleration of the ownship
$r^0$ : Rate of change of acceleration of the ownship
$d^{\text{LOS}}$ : Loss of separation distance between the ownship and the intruder
$d_{\text{goal}}^i$ : Distance to the intruder's goal
$v^i$ : Velocity of the intruder
$a^i$ : Acceleration of the intruder
$r^i$ : Rate of change of acceleration of the intruder
$d_0^i$ : Initial distance between the ownship and the intruder
$d_{\text{int}}^0$ : Initial relative distance between the intruder and the ownship
$d_{\text{int}}^i$ : Current relative distance between the intruder and the ownship

### 3.3. Action space

In the MARL model, the objective is to provide suitable actions, specifically speed advisories, to the AAM vehicle to prevent conflicts during the flight. There are three possible actions that any vehicle on the same route can take: accelerating, decelerating, or maintaining the cruise speed (no acceleration). To manage the complexity of decision-making, we impose a constraint where each agent is allowed to choose an action every 5 s. This limitation ensures safe separation for the AAM vehicle considered as a case study in this paper. The action space is defined as follows:

$$A_t = [V_-, 0, V_+]$$

where $V_-$ means the vehicle will decelerate, 0 denotes no acceleration or the vehicle will maintain the current speed, and $V_+$ means that the vehicle will accelerate. It is important to note that our focus is on action

selection within this discrete action space, represented as $A_t$, rather than specifying the exact magnitudes of real-world acceleration. The goal of our MARL model is to provide suitable action recommendations (speed advisories) to the AAM vehicle to prevent conflicts during flight while considering these discrete action choices.

### 3.4. Reward function

The reward function comprises two components: an immediate reward and a terminal reward. The immediate reward, given to the agent at each time step, is proportional to the difference between the current separation distance and the minimum safe separation distance. The terminal reward is given to the agent at the end of the episode, which is based on the total flight time and the final separation distance from other agents. The agent's goal is to maximize the cumulative reward, which represents the trade-off between safety and efficiency.

We formulate identical reward functions for all agents, where each agent aims to maximize its local reward. In case of a conflict between two agents, a penalty is imposed to account for the negative impact resulting from the conflict. A conflict is defined when the distance between two aircraft is less than two nautical miles (i.e., $d^{\mathrm{LOS}} = 2$ NM), which is a parameter setting for the aircraft model we use in BlueSky for demonstration purposes and can be adjusted for AAM missions. The reward function utilized for the actor-critic network is defined as follows:

$$R_t = \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} + \gamma^n V_\phi(r_{t+n+1}) \tag{20}$$

$$r_t = \begin{cases} -1 & \text{if} \quad d^{\mathrm{LOS}} < 2 \text{ NM} \\ \epsilon & \text{if} \quad 2 \text{ NM} \le d^{\mathrm{LOS}} \le 4 \text{ NM} \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

where the discounted sum of the next $n$ rewards $\sum_{k=0}^{n}\gamma^k r_{t+k+1}$ is computed for each state encountered, and the value of the state $V_\varphi(r_{t+n+1})$ encountered $n$ steps later is estimated by the critic network. $R_t$ is the total return; $\epsilon$ is some small negative reward when the vehicle gets to the conflict distance; and $\gamma(0 \le \gamma \le 1)$ is the discount factor, which determines the present value of future rewards: a reward received $k$ time steps in the future is worth only $\gamma^{k-1}$ times what it would be worth if it were received immediately. The agent will be trained to optimize its performance and achieve higher rewards based on our model.

The terminal state is defined as the state in which all agents have successfully reached their designated target positions. The structure of the terminal state reward is defined as follows:

- + 25 when all the vehicles successfully reach their target positions;
- − 5 when a vehicle has not yet reached its target position;
- − 10 when the vehicles collide along the way.

Each agent's objective is to maximize its long-term performance and accumulate long-term rewards, even though feedback is provided only based on the one-step performance at a time. By combining the A2C and PPO algorithms, it is possible to achieve the optimal policy. The Actor component is responsible for the policy $\pi$ and is utilized for selecting the agent's actions and updating the policy network accordingly. The Critic component corresponds to the value function $Q(s, a)$ for action value or $V(s)$ for state value. The actor-network receives an observation (e.g., state) as input and generates a probability distribution for selecting actions. Sampling from this distribution allows the selection of an action. The critic-network also takes the state as input and outputs a single integer representing the estimated value of that state, which serves as an approximation of the state value function.

Gradient descent is used to optimize the objective function in policy gradient techniques. Initially, we define the probability ratio $r(\theta)$ between the old policy $\pi_{\theta_{\mathrm{old}}}$ and the new policy $\pi_\theta$ using the following definition:

$$r_t(\theta) = \frac{\pi_\theta(s|a)}{\pi_{\theta_{\mathrm{old}}}} \tag{22}$$

where $\theta$ represents the weights of the neural network. The probability ratio $r(\theta)$ indicates that when $r(\theta) > 1$, the action is more likely to be selected in the new policy, whereas when $r(\theta) < 1$, the action is less likely to be chosen. Based on this probability ratio, a simple objective function can be formulated as follows:

$$L^{\mathrm{CPI}}(\theta) = E_t\left[\frac{\pi_\theta(s|a)}{\pi_{\theta_{\mathrm{old}}}} A_t\right] = E_t[r_t(\theta)A_t] \tag{23}$$

In the PPO algorithm, a truncated version of the Generalized Advantage Function (GAE) [47] is used, which is defined as follows:

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots\ldots\ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{24}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$.

When the action selection probability is low with the old policy, the objective function $L^{\mathrm{CPI}}(\theta)$ can be unstable because $r_t(\theta)$ is high, causing the updates to be enormously high. PPO recommends two solutions: PPO-Clip and PPO-Penalty [48]. We use PPO-Clip since it is the most prevalent. To minimize the number of updates, PPO-Clip simply clips the probability ratio $r_t(\theta)$. The ratio must fall within a certain interval $(1 - \epsilon, 1 + \epsilon)$, where $\epsilon$ is a clipping hyperparameter. After that update, the objective function becomes:

$$L^{\mathrm{CLIP}}(\theta) = E_t[\min(r_t(\theta)A_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t] \tag{25}$$

Since the objective function in PPO only utilizes the policy (actor) network, an additional term is required to update the critic network. The role of the critic network is to minimize the discrepancy between the estimated and actual values. By employing squared loss, the objective function for the critic network can be expressed as:

$$L^{\mathrm{VP}}(\theta) = (V_\theta - V^{\mathrm{Target}})^2 \tag{26}$$

Furthermore, to encourage exploration, an entropy term is incorporated into the objective function. The final objective function can be defined as follows:

$$L(\theta) = \left(L^{\mathrm{CLIP}}(\theta) - c_1 L^{\mathrm{VP}}(\theta) + c_2 S[\pi_\theta](s)\right) \tag{27}$$

## 4. Network design and hyperparameter tuning

In the previous section, we presented the formulation of the merging and intersection problems within the air corridor as an MARL problem. We also established the reward structure of the problem, which guides the agent in seeking the optimal policy. In this section, we proceed with the development of a solution to this problem utilizing neural networks and conduct an investigation into the impact of hyperparameters on the algorithm's performance.

### 4.1. Network design

To train and evaluate our MARL model, we use a single neural network that distributes optimal speed advisories to each agent, ensuring conflict-free flights at merging points and intersections. In order to formulate this environment as an MARL problem, we adopt a centralized learning, decentralized execution (CLDE) approach, employing a shared neural network for all agents. Through this approach, the MARL model is trained to facilitate cooperation among all agents by utilizing the shared network. While the policy-based MARL model is formed as CLDE that helps the agent to learn their local policies [49] and all the agents share the same neural network, their actions can still be different in the execution. Fig. 4 shows the MARL neural network architecture. The shared neural network can be implemented in all AAM vehicles to ensure

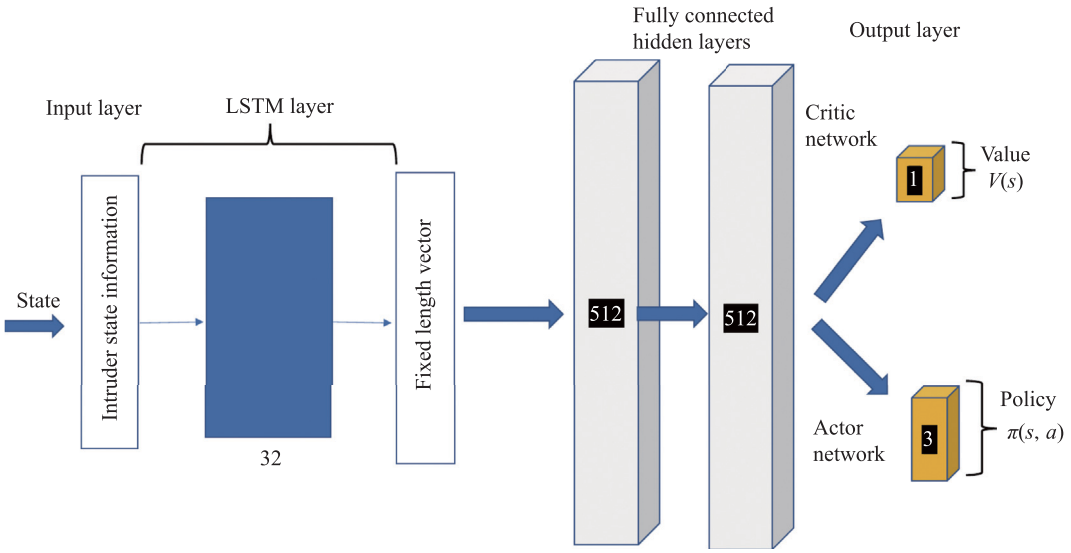



**Fig. 4.** MARL neural network architecture.

safe separation during flight, eliminating the need for individual neural networks for each vehicle.

The first layer of the network is an LSTM layer that encodes all the intruder information into a fixed-length vector. Subsequently, this information is passed through two fully connected layers to generate outputs for the policy and value function. Each vehicle then follows this policy until the termination of the episode. As the environment is stochastic, a single episode's policy may not provide clear decision-making guidance for each agent. Therefore, by collecting multiple episodes, the neural network policy can be updated, allowing the network to produce different outcomes from the same policy.

Our chosen reinforcement learning algorithm is a combination of A2C and the PPO loss function, which is a policy-based approach. We utilize a shared layer between the actor and the critic in our network architecture. In this network, we utilized two identical layers, each consisting of 512 nodes, along with an LSTM encoder with 32 nodes. The activation functions used in our neural networks are the rectified linear activation function (ReLU) for the hidden layers and the hyperbolic tangent function (tanh) for the LSTM layer. ReLU is a widely used activation function in neural networks due to its performance and ease of training. The softmax activation function is applied to the output of the actor's network, as it scales numbers into probabilities, while the linear activation function is used for the output of the critic's network. The softmax activation function is commonly used as the final activation function in neural networks [50]. At the end of the network, two fully connected layers (hidden layers) generate the policy and value outputs for the given state. The policy is initialized at the beginning of each episode and updated at the end of each episode.

The neural network design allows us to update the policy of each agent at the start of each episode. Each AAM vehicle then follows its individual policy until it reaches its destination. The key parameters for the model include a learning rate of 0.000,1, a PPO ratio bound of 0.2 (represented as $\epsilon$), a discount factor of 0.99 (represented as $\gamma$), a reward coefficient of 0.1 (represented as $\alpha$), a reward coefficient of 0.005 (represented as $\delta$), and an entropy coefficient of 0.000,1 (represented as $\beta$). These parameters play a crucial role in determining the performance and convergence of the MARL model.

To further enhance the capability of our MARL model and to ensure that we capture relevant patterns in the intruder information efficiently, we experimented with the attention mechanism as an alternative to the LSTM layer. Attention mechanisms [51] have been revolutionary in several machine learning tasks due to their ability to dynamically focus on different parts of the input sequence, highlighting relevant features while downplaying others. This potentially offers a way to better encode the intruder information by emphasizing critical parts that can influence the policy and value predictions. The intruder data is first processed via a fully connected layer, then passed to an attention network which produces a weighted, fixed-length vector. This vector is concatenated with the ownship state data and sent through two fully connected layers of 512 nodes each. The final layer, consisting of 4 nodes, generates the policy and value outputs, ensuring our model captures relevant details while making efficient decisions.

We designed an alternative MARL neural network architecture where the LSTM layer was replaced with a self-attention mechanism. The attention mechanism dynamically weighs the sequence of intruder information and produces a fixed-length vector, just as the LSTM layer does. This encoded information was then passed through the subsequent fully connected layers, identical to the original architecture.

After training models using both the LSTM and attention mechanisms, maintaining uniform conditions such as identical hidden layers and activation functions, we embarked on a performance comparison. From Figs. 5 and 6, it is evident that the LSTM model (green line) converges faster and demonstrates a higher success rate (fewer collisions) compared to the attention-based model. This underlines the superior performance of the LSTM layer in ensuring safe and efficient paths for the agents in our MARL framework. Even though the attention mechanism brought a sophisticated approach to encoding input sequences, the LSTM layer was distinctly superior in our particular context. Remarkably, the LSTM-based model not only converged more swiftly but also exhibited superior proficiency in orchestrating conflict-free flights, especially at crucial merging points and intersections.

Several factors illuminate the edge of LSTM in our framework. LSTMs, with their recurrent nature, are intrinsically tailored to map temporal dependencies within sequences. In contrast, attention mechanisms, while offering a broader perspective by prioritizing diverse segments of an input sequence, can occasionally overlook the temporal intricacies pivotal to our application's challenges. Additionally, the LSTM's architectural design, characterized by its gating mechanisms, excels at preserving long-term dependencies, proving especially potent for the dynamics of our MARL setting.

In conclusion, while attention layers have shown success in many areas, our tests emphasize the importance of choosing the right model structure for the specific problem. In our MARL framework, LSTMs clearly stood out, converging faster and ensuring safe, conflict-free paths for agents in short time.

In the upcoming subsection, we will provide a detailed explanation of how the hyperparameters were tuned. Once the parameters are set, the

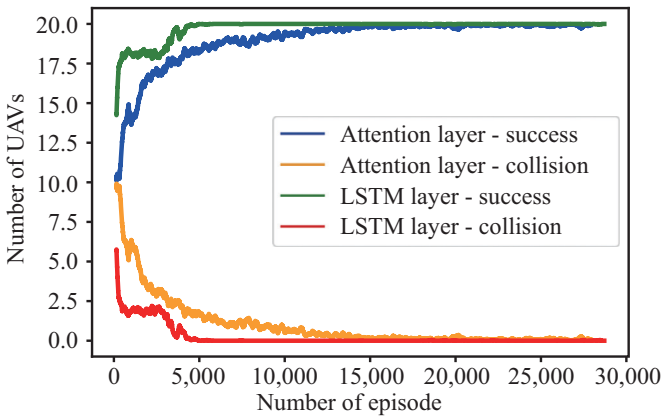



**Fig. 5.** Performance comparison of MARL models: LSTM vs. attention layers.

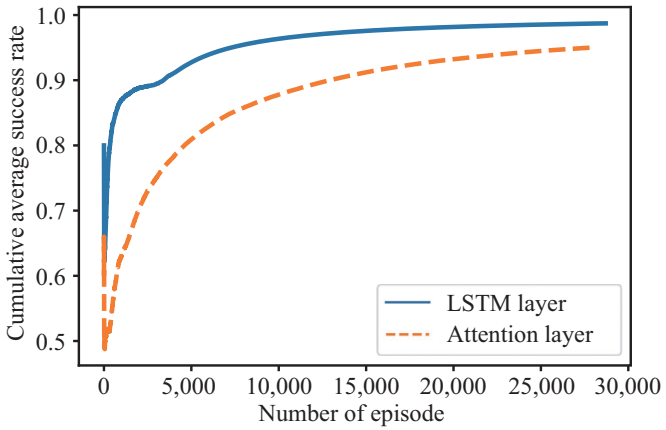



**Fig. 6.** Cumulative success rate: LSTM vs. attention layers.

model is trained over 10,000 episodes, allowing it to learn different scenarios in the environment. With the utilization of a single network, each AAM vehicle is able to determine its own speed at the merging point or intersection by following the optimal policies. The ultimate objective of our research is to achieve conflict-free flights for every AAM vehicle. In our case studies, we focus on a single merging point and intersection scenario. However, it is expected that our model can be extended to handle multiple merging points and intersections within the same air corridor.

### 4.2. Hyperparameter tuning

Hyperparameter tuning is a process of selecting the best set of hyperparameters for a reinforcement learning agent. The values of hyperparameters, such as the learning rate or discount factor, are set prior to training and determine the learning process. Finding the optimal hyperparameters for a reinforcement learning agent requires performing experiments with different values to see how they impact the performance. In MARL, the same process applies, but for multiple agents. The hyperparameters for each agent may be different, and finding the best hyperparameters for each agent can be a challenging task. The selection of a model's hyperparameters can affect how long it takes to train and test the model design.

Hyperparameter tuning in the context of AAM traffic control and management using reinforcement learning involves finding the optimal values for the parameters that control the behavior of the reinforcement learning algorithm. The goal is to achieve efficient and safe traffic control in airspace. Some important hyperparameters to tune in this context include: 1) learning rate that determines the step size of weight updates in the reinforcement learning algorithm; 2) discount factor that determines the relative importance of future rewards in the reinforcement learning algorithm; and 3) exploration rate that controls the level of exploration vs. exploitation in the reinforcement learning algorithm.

Hyperparameter tuning can be done using various methods such as grid search [52], random search [53], or Bayesian optimization [54]. The choice of method will depend on the specific requirements and constraints of the AAM traffic control problem. While Bayesian optimization is a powerful method for hyperparameter tuning, in our specific implementation, we opted for a sample-based approach to determine suitable values for the hyperparameters. We conducted a series of experiments using different combinations of hyperparameter values and evaluated their impact on the performance of the AAM traffic control model.

Learning rate is an important hyperparameter to tune in reinforcement learning. The learning rate determines how quickly the agent updates its policy in response to new information. A high learning rate will result in rapid updates, while a low learning rate will result in slower updates. Finding the right balance between learning too quickly and learning too slowly is crucial for the agent's performance. Also, the optimal learning rate depends on the specific reinforcement learning problem and the size and complexity of the state space and action space. Specifically, a small learning rate is typically used when the reinforcement learning problem is complex, the environment is non-stationary, or the agent is dealing with a high dimensional state or action space. This helps to ensure a stable and robust learning process, even if the convergence may be slower. On the other hand, a big learning rate can cause the learning process to converge faster, but may also result in instability or divergence of the algorithm. When a big learning rate was used, overshooting may occur because the algorithm updates the weights too quickly that may cause it to oscillate. Big learning rates are typically used when the reinforcement learning problem is simple, the environment is stationary, or the agent has a clear understanding of the reward function. In these cases, the learning process can converge faster, allowing the agent to learn quickly and make better decisions.

In our MARL model, we use different learning rates to find out the best fit for our model to reduce learning instability, as shown in Fig. 7, where RMS represents the running model success and RMC represents the running model collision. As can be seen from Fig. 7, the choice of learning rate has a significant impact on the performance of the learning algorithm. A small learning rate means that the weight updates in the learning process will be slow, which can lead to a more stable but slower convergence. Based on these results and analysis, the learning rate is selected as LR = 0.000,1 in our proposed MARL setting to deal with the complexity of the problem and the high dimensional state or action space.

The discount factor is also an important hyperparameter that needs to be tuned in reinforcement learning. The discount factor determines the

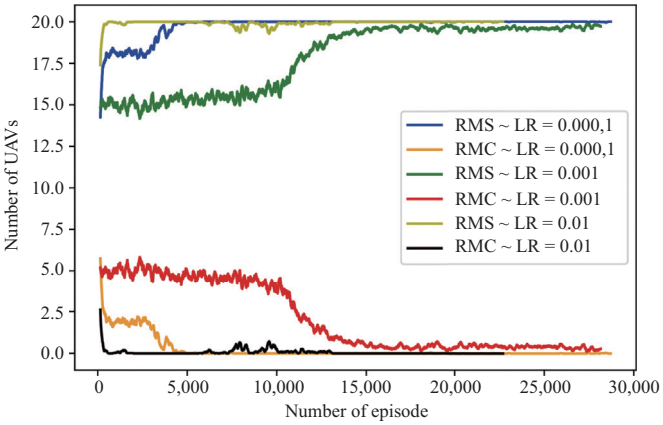



**Fig. 7.** Learning rate for MARL model performance.

relative importance of future rewards in the reinforcement learning algorithm. A high discount factor means that future rewards are valued more than immediate rewards, while a low discount factor means that immediate rewards are valued more. In the context of AAM traffic control using reinforcement learning, the choice of discount factor depends on the specific problem formulation and the trade-off between immediate and long-term rewards. If the goal is to reduce congestion in the airspace, a high discount factor may be appropriate, as reducing congestion may require taking actions that result in delayed rewards. On the other hand, if the goal is to ensure safety, a low discount factor may be more appropriate, as immediate rewards (e.g., avoiding collisions) are more important. To determine the appropriate discount factor for the problem, a range of discount factor values should be tested and their impact on the performance of the reinforcement learning algorithm should be evaluated. The choice of discount factor should be based on a balance between the long-term goals of the AAM traffic control system and the need for immediate actions that ensure safety. Different values of the discount factor, such as 0.1, 0.5, 0.9, and 0.99 have been tested to see how they impact the performance of the MARL algorithm.

Fig. 8 shows the learning curves for our reinforcement learning algorithm with different discount factors (i.e., 0.1, 0.5, 0.9, and 0.99). The *x*-axis represents the number of episodes, and the *y*-axis represents the performance metric (e.g., cumulative reward). Each line in the figure represents the learning curve for a specific discount factor. The learning curve for a discount factor of 0.1 starts with a steep increase in performance; however, it quickly plateaus and remains at a low level for the rest of the training. The learning curve for a discount factor of 0.5 shows a more gradual increase in performance, with a slower initial improvement but eventually achieving better performance than a discount factor of 0.1. The learning curve for a discount factor of 0.9 shows an even slower initial improvement, but performs better than a discount factor of 0.5. Finally, the learning curve for a discount factor of 0.99 leads to the slowest initial progress, but it eventually achieves the best performance of all the discount factors. In sum, Fig. 8 illustrates how the choice of discount factor can impact the performance of the proposed MARL algorithm and highlights the trade-off between immediate and long-term rewards.

Exploration rate is another important hyperparameter that needs to be adjusted in the MARL algorithm, particularly in the setting of AAM traffic control. A high exploration rate indicates that the agent is willing to try new states and actions, even if it means foregoing short-term rewards. This can be helpful when the agent is first becoming familiar with the environment and needs to explore different possibilities to develop a good policy. On the other hand, a low exploration rate indicates that the agent is more likely to exploit short-term rewards by taking advantage of its current knowledge, even if it means giving up on possible long-term rewards. This can be useful when the agent has already learned a good policy and wants to optimize its performance.

In the MARL setting, there are multiple agents that interact with each other and the environment, which can make the learning process more complex. The exploration strategy used by each agent can affect the overall performance of the system. In our experiment, we have explored different exploration rates to find the optimal value for our MARL model. We have tested a high exploration rate of 1.0, a decaying exploration rate that started at 0.5 and decayed by a factor of 0.99 every 1,000 time steps, and a low exploration rate of 0.1. Decaying exploration rate is a useful strategy for MARL for balancing exploration and exploitation. Decaying exploration rate is a strategy that starts with a high exploration rate and gradually decreases it over time as the agent becomes more familiar with the environment. The idea behind this strategy is that the agent needs to explore different actions and states early on in the training process to learn about the environment and build a good policy. As the agent gains more knowledge and experience, it can shift towards exploiting its current knowledge to maximize rewards.

The performance of our MARL model with different exploration rates is shown in Fig. 9, from which we can see that the high exploration rate initially performed well, but its performance shortly plateaued and remained relatively low throughout training. The low exploration performed better than high exploration rate, but its performance still unstable and fluctuated over time. The decaying exploration rate, on the other hand, started with low performance in the beginning with high exploration and gradually shifted towards exploitation as the agent gained more knowledge that is leading to more stable and consistent performance over time. The experiments showed that the decaying exploration rate approach resulted in the best overall performance, and we therefore chose the decaying exploration rate for our model.

### 4.3. Performance evaluation: A2C vs. DDPG

In the complex world of autonomous vehicle operations, selecting the right reinforcement learning algorithm plays a pivotal role in achieving efficient and safe vehicular behaviors. To elucidate the capabilities and limitations of contemporary algorithms, we employed two widely recognized reinforcement learning methods, A2C and DDPG, to a vehicle merging scenario, which is emblematic of real-world complexities and challenges. Given the dynamic nature of this problem, which involves multiple agents (in our case, 10 vehicles) with intertwined actions and outcomes, achieving optimal policies while ensuring safety is paramount. Our experiments spanned over 30,000 episodes as shown in Fig. 10, and for a fair evaluation, we ensured that both algorithms utilized identical network architectures. The intent was to delve deep into the nuances of each algorithm's learning curve, rate of convergence, and overall performance, while isolating the algorithm as the primary variable of interest.

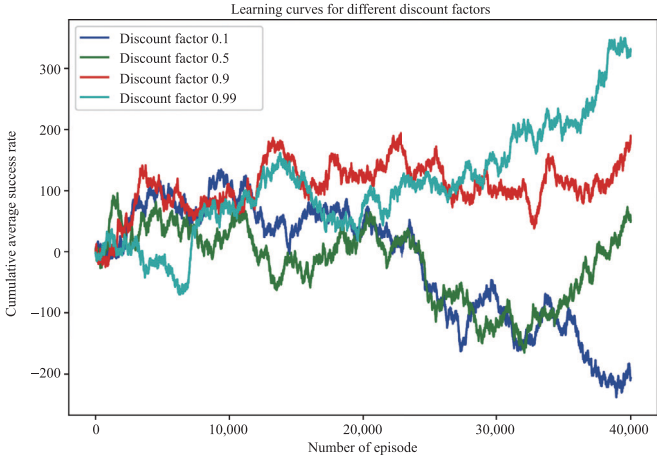



**Fig. 8.** Learning curves for different discount factors.

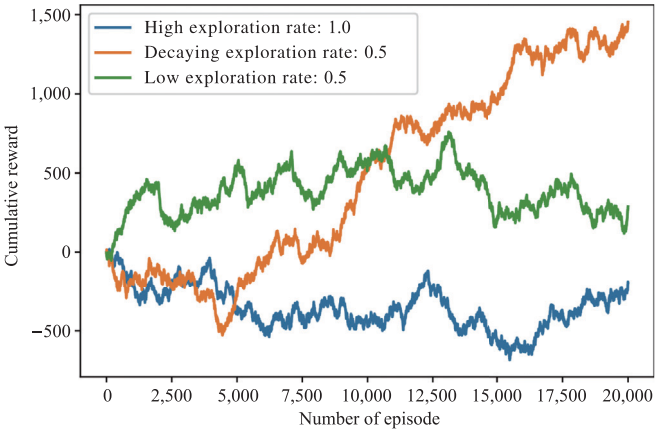



**Fig. 9.** Cumulative reward over time for different exploration rates.

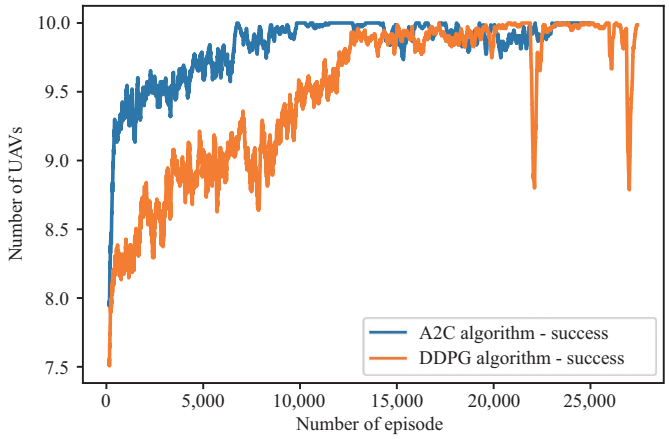



**Fig. 10.** Comparison of learning curves for A2C and DDPG.

Training time and efficiency: from a time-efficiency perspective, there was a notable difference between the two algorithms. Training the A2C model took approximately 96 h, equivalent to 4 days. In contrast, the DDPG model required around 126 h to train for the same number of episodes, a significant 31% increase in training time. Convergence rate: regarding the learning curves and convergence rates, A2C outperformed DDPG. The A2C model demonstrated a smoother learning curve and reached optimal policies at a faster pace. DDPG, on the other hand, displayed initial challenges, particularly during the early phases of training. The convergence for DDPG was observed around 10,000 episodes, which is notably slower than A2C. Safety and accidents: safety is paramount in vehicle merging scenarios. While A2C showed superior performance in this regard, DDPG struggled initially, recording a higher number of accidents during the early training episodes. Even post-convergence, DDPG still sporadically experienced accidents, indicating potential instability in its policy updates or exploration mechanisms.

In sum, in the merging scenario involving 10 vehicles, A2C demonstrated superior reliability and efficiency compared to DDPG. While A2C exhibited faster convergence and a smoother learning curve, indicating an enhanced capability to optimize in the environment, DDPG faced initial challenges and a longer training duration. The observed disparities might stem from the intrinsic differences between the algorithms, particularly DDPG's deterministic policy gradient in contrast to A2C's stochastic nature. Consequently, for this specific scenario, A2C emerged as the more effective and reliable reinforcement learning approach.

### 4.4. Algorithm performance by choosing random sample

In reinforcement learning, the mean, median, and midrange of a sample can provide some insight into the learning process. The mean is the average of the sample, calculated as the sum of the values divided by the number of values. The mean is a useful measure of central tendency, as it provides a single number that summarizes the distribution of the sample. However, it is sensitive to outliers and can be misleading if the sample is not normally distributed. The median is the middle value of the sample, calculated by arranging the values in order and selecting the middle value (or the average of the two middle values if the sample size is even). The median is a robust measure of central tendency, as it is not sensitive to outliers. It is especially useful in reinforcement learning when the sample distribution is not normal. The midrange is the average of the minimum and maximum values of the sample. The midrange provides a simple measure of the spread of the sample and can be useful for determining the range of values in the sample.

By calculating these statistics, one can get an idea of the distribution of the sample and how the learning process is progressing over time. For example, the mean or median can be used to track the progress of the reinforcement learning algorithm and monitor how the reward signal changes over time. Similarly, the midrange can provide information on the range of values in the sample, which can be useful for detecting outliers or unusual events in the learning process.

Figs. 11 and 12 show the performance of the MARL algorithm based on the mean, median, and midrange statistics values. The mean value of the success is measured as 9.6 out of 10, while the median is 9.7 and the midrange is 9.4. The mean and median provide a measure of the typical reward received by the agent. High values of the mean and median in reinforcement learning typically indicate that the agent is performing well and receiving high rewards for its actions. This may be a result of the agent learning an optimal policy or a good strategy for performing the task.

It is important to note that a high value for the mean and median does not guarantee that the agent has learned an optimal policy. There may be other factors that contribute to high rewards, such as a favorable initial state or a lucky sequence of actions. To ensure that the agent has learned an optimal policy, it may be necessary to evaluate its performance on a held-out test set or by comparing it to other algorithms. In summary, a high value of the mean and median in reinforcement learning is generally a positive sign that the agent is performing well and receiving high rewards, but further evaluation may be necessary to ensure that the agent has learned an optimal policy.

### 4.5. Experiment on method efficiency

In this section, we present the results of our experiments focused on evaluating the conflict avoidance rate in merging case and intersection case scenarios involving 20 AAM vehicles. This result is instrumental in assessing the effectiveness of our MARL approach, shedding light on its capacity to manage high-density AAM operations while ensuring safety and efficiency.

The conflict avoidance rate as shown in Figs. 13 and 14 for merging and intersection cases, which serves as a crucial indicator of the safety and effectiveness of our MARL-based AAM management approach, is a key focus of our analysis. It measures the percentage of conflicts successfully avoided during AAM operations. In the intricate merging and intersection scenarios featuring 20 AAM vehicles, our MARL model demonstrated exceptional performance, achieving a conflict avoidance rate of 99%. This remarkable outcome underscores the capability of our approach to mitigate potential conflicts, enhance the safety of AAM operations in congested airspace, and instill confidence in the feasibility of autonomous, high-density air transportation.

In conclusion, the outcomes of our experiments in both scenarios involving 20 AAM vehicles provide compelling empirical evidence of the effectiveness of our MARL approach. The outstanding conflict avoidance rate underscores the capabilities of our model in managing high-density AAM operations while prioritizing safety. These findings align with the broader goals of AAM, emphasizing the potential of advanced learning-based algorithms to transform urban, suburban, and rural air transportation into a safer and more efficient mode of travel.

## 5. Simulation results

Deep reinforcement learning agents need to undergo training in an environment that allows them to learn and develop decision-making capabilities. To evaluate the performance of our MARL model, we use the BlueSky air traffic control simulator, which is an open-source and open-data multi-platform simulation tool [14]. BlueSky has been used in previous research to demonstrate reinforcement learning algorithms and assess the performance of MARL models. Although the BlueSky simulation environment does not include an eVTOL vehicle model, we conduct our experiments using the fixed-wing Airbus A318 aircraft. Two specific case scenarios are created within the BlueSky environment to evaluate the effectiveness of our proposed MARL approach.

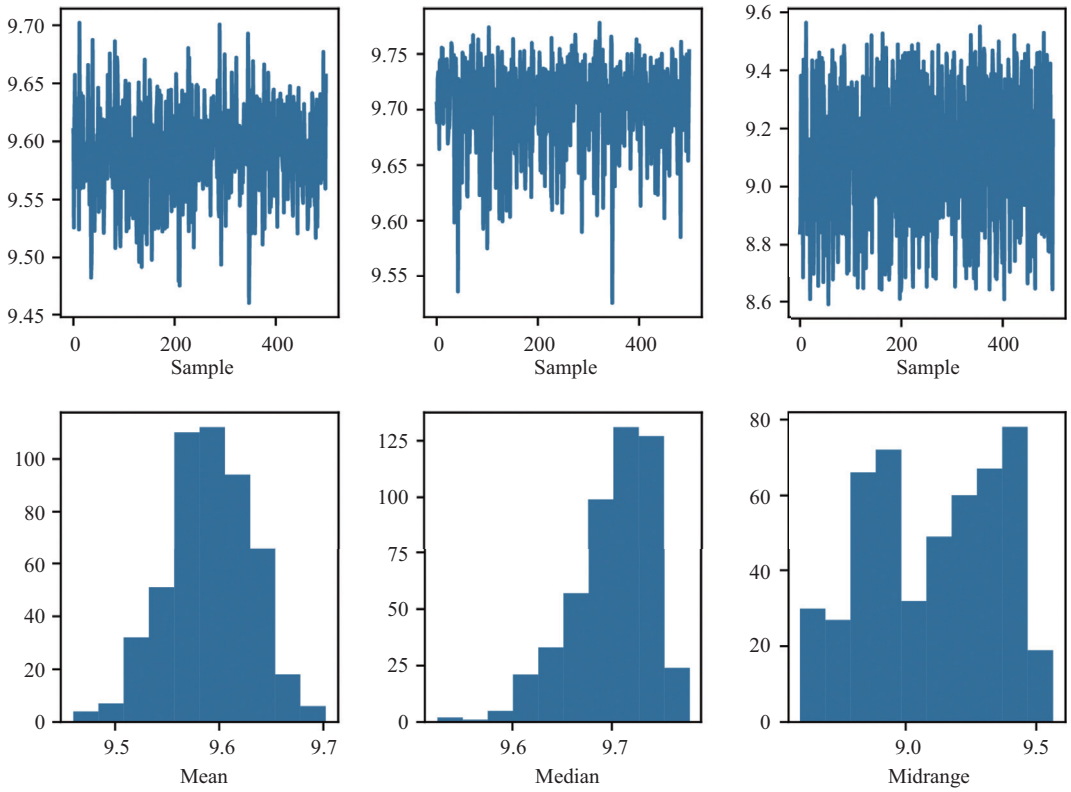



**Fig. 11.** Performance of MARL algorithm based on mean, median, and midrange statistics for success of random training sample.

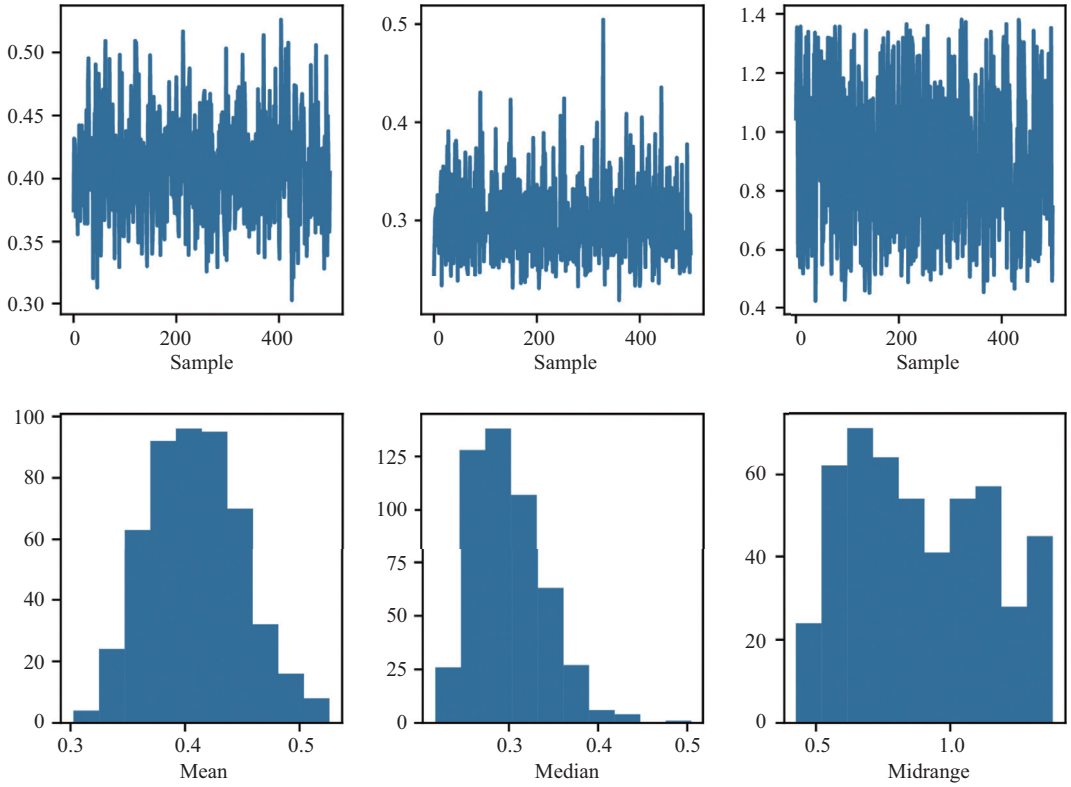



**Fig. 12.** Performance of MARL algorithm based on mean, median, and midrange statistics for collision of random training sample.

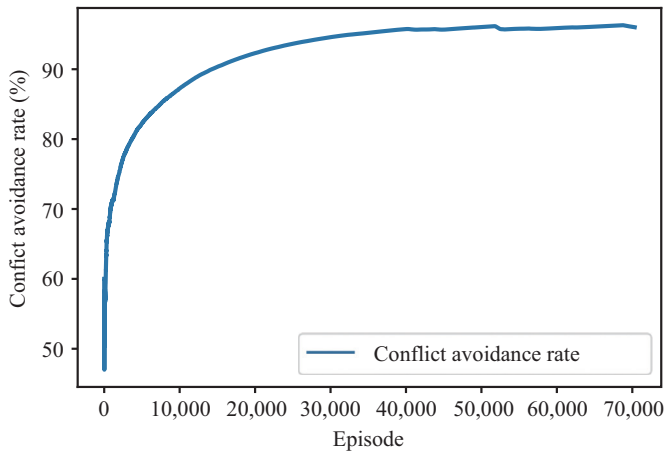



**Fig. 13.** Conflict avoidance rate in merging case scenario.

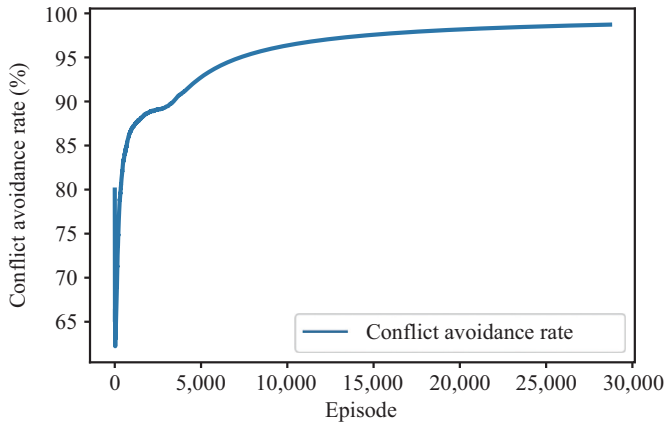



**Fig. 14.** Conflict avoidance rate in intersection case scenario.

### 5.1. *Merging scenario*

It is undeniable that the coordination between vehicles is a critical enabler for safe and effective merging maneuvers. While merging is relatively easy for fully autonomous ground vehicles in a smart surface transportation environment, coordination of air vehicles in the AAM environment is a significantly more challenging task due to the complex scenarios and a lack of communication between vehicles. In fact, autonomous traffic control and management at merging airways has been deemed as one of the main challenges for AAM operations. The vehicles on the merging airway needs to efficiently merge into the main air corridor without collision. In a cooperative environment, the AAM vehicles on the main corridor should proactively slow down or speed up to create enough room for the vehicles on the merging corridor to join safely. The merging vehicles should also adjust their speeds quickly to join the main corridor when it is safe to prevent the collision. With our proposed data-driven methods, the communication between the AAM vehicles is assumed to be maximized for safe merging operations.

Fig. 15 illustrates the merging case scenario in the BlueSky simulation environment. The horizontal lane symbolizes the primary corridor, whereas the adjacent lane represents the merging lane or the secondary aerial corridor. The aircraft along the route are symbolized by green triangle shapes, while the airports on the map are represented by white triangle and square shapes. Our case study focuses on generating an optimal policy for the main aircraft (ownship) to safely merge with other aircraft without causing conflicts at the merging point. To achieve this goal, we define a reward structure that considers the ownship's speed, acceleration, and distance to the merging point and other aircraft. The policy-based MARL model is implemented using the CLDE approach, where all agents share the same neural network and the policy is learned through cooperation.

Each simulation run in the BlueSky is referred to as one episode. For this case study, each episode comprises 10 aircraft that enter the airspace randomly. This aircraft arrival pattern or aircraft entry sequence is utilized for training the model. In our training process, each aircraft is assigned its own objective to accomplish within the environment. We conduct 100,000 episodes to train the agents and evaluate the performance of our model. Through the training, we observe that each agent is able to successfully reach its destination without causing any conflicts. In each episode, the initial positions, initial speeds, and destinations of the aircraft are randomly generated within the BlueSky environment. It is important to note that training and testing our model with a higher number of vehicles or increased air traffic density would yield more accurate results for AAM operations. However, due to the limitations of available aircraft models in the BlueSky environment, we specifically test our model using the Airbus A318, which is one of the smaller aircraft in the simulation, as an example to showcase the effectiveness of our approach.

We evaluate our MARL model's performance in two different merging case scenarios with varying numbers of aircraft (i.e., different aircraft density levels). The first case scenario involves 10 aircraft, while the second case scenario involves 50 aircraft.

Fig. 16 shows the results of 40,000 episode runs with 10 aircraft, where we can see that the model behavior gets better over time. While the orange curve represents the number of collisions, the blue curve shows the successful pass at the merging point. After 10,000 episode runs, our model starts a better performance and generates better policies that lead each agent to safe separations and merging. In Fig. 17, training of the model with 10 vehicles over 100,000 episodes clearly demonstrate the model performance and test all the possible merging cases in our simulation. As can be seen from Fig. 17, the performance of the model can be evaluated by the mean value of the model's success and collision at the end of the episode. The proposed method consistently outperforms the benchmark in all traffic levels during the

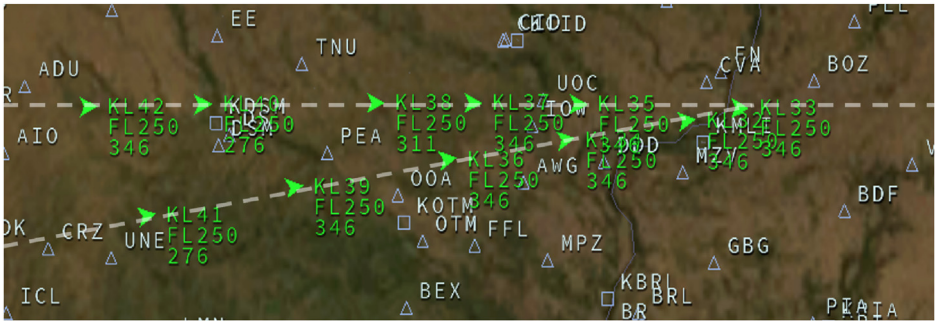



**Fig. 15.** Merging control case study in BlueSky environment.

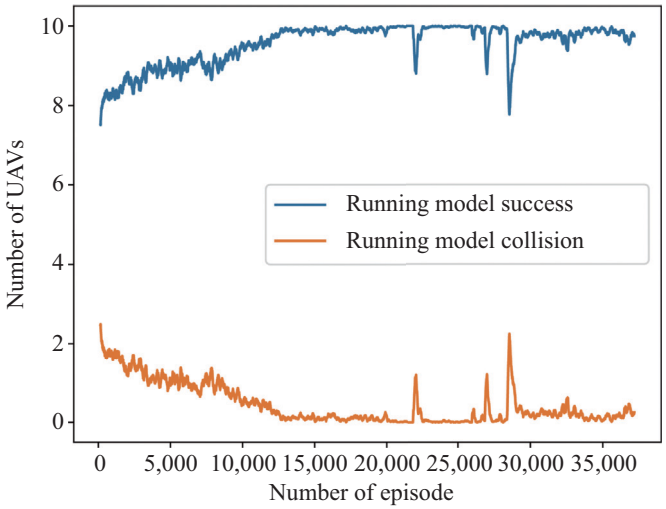



**Fig. 16.** MARL model training with 10 vehicles.

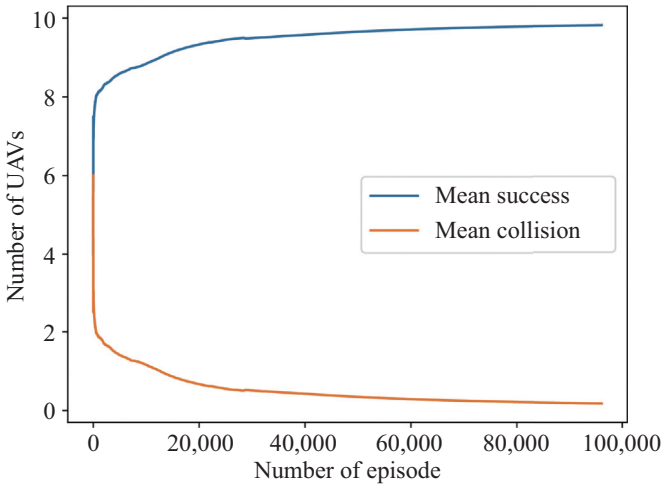



**Fig. 17.** MARL model performance with 10 vehicles.

merging. The model can learn the optimal policy for the ownship to safely merge with other aircraft, and the results show that the average speed and acceleration of the ownship are within the safe limits during the flight.

To further assess the performance of the merging algorithm in the context of different aircraft density levels, we run the simulations with 50 vehicles and record metrics such as the distance between the vehicles and any conflicts or collisions that occur during the merging process, as shown in Figs. 18 and 19. The results suggest that the proposed data-driven air traffic management system is capable of effectively managing complex air traffic scenarios and maintaining safe separation between aircraft, even under high aircraft density levels. Testing and development in practical real-world implementations are necessary to ensure the system's reliability and efficacy. Additionally, the algorithm's performance may depend on other factors such as the merging point's geometry, vehicle size and speed, and external factors like weather and structural constraints of the AAM system. Therefore, comprehensive testing under various conditions is essential to fully evaluate the algorithm's performance in future studies.

### 5.2. Intersection scenario

The intersection is another important AAM scenario in which two or more AAM vehicles cross each other's path at a point in the air. Like the

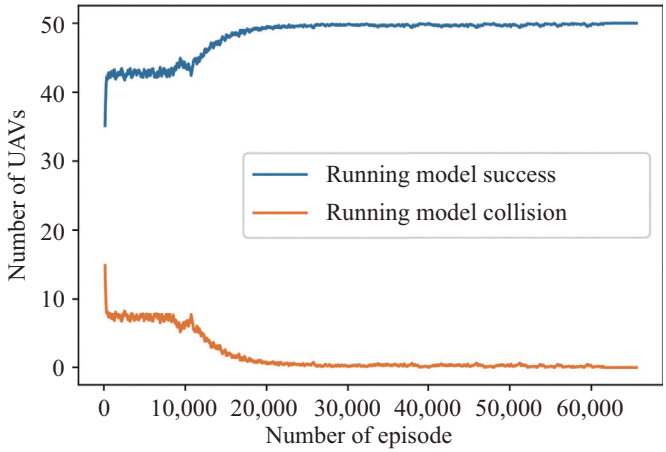



**Fig. 18.** MARL model training with 50 vehicles.

merging scenario, the intersection scenario also requires safe separation between the vehicles to prevent any collision. In the intersection scenario, the state includes information about the position and dynamics of each vehicle. The actions in this scenario are the speed adjustments of each vehicle, and the reward is defined based on the minimum safe separation distance between the vehicles. The reinforcement learning model's objective is to adjust the speed of each vehicle to ensure safe separation while passing the intersection.

An intersection case scenario is created in BlueSky as shown in Fig. 20, and the parameters for the intersection case are modified. The simulator allows us to observe the interactions between vehicles and evaluate the performance of our MARL model in terms of safety, efficiency, and fairness. We modify the parameters for the intersection case and test two different aircraft density levels: 10 aircraft and 20 aircraft. Figs. 21–24 show the performance of the intersection scenario for two density levels. Our proposed algorithm effectively avoids conflicts by providing speed advisories to each vehicle, ensuring safe and efficient intersection control for AAM.

In the first intersection case study, each episode consists of 10 randomly entering aircraft in the airspace for model training. Our MARL model undergoes 40,000 episode runs to train the agents and evaluate its performance. By the end of the training, we observe that each agent successfully reaches its destination and passes the intersection without conflicts. Fig. 21 shows the results of 30,000 episode runs for 10 vehicles, illustrating the gradual improvement in model behavior over time. The orange curve represents the number of collisions, while the blue curve shows the successful pass at the intersection. After 5,000 episode runs,

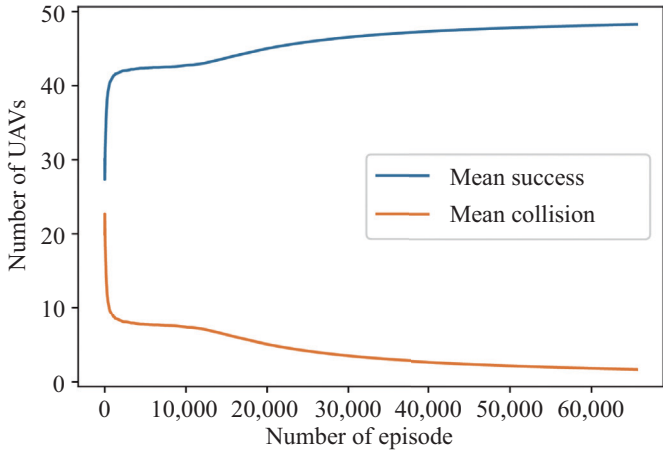



**Fig. 19.** MARL model performance with 50 vehicles.


**Fig. 20.** Intersection case study in BlueSky environment.

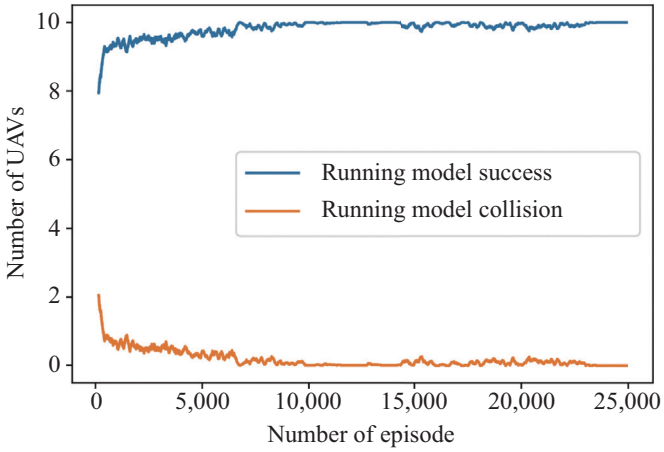



**Fig. 21.** MARL model training with 10 vehicles.

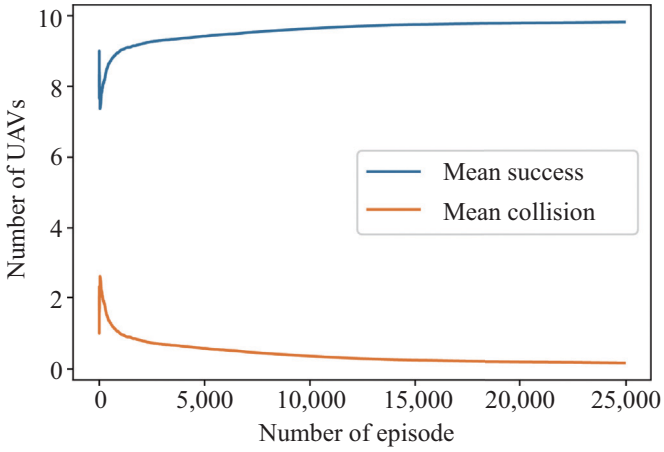



**Fig. 22.** MARL model performance with 10 vehicles.

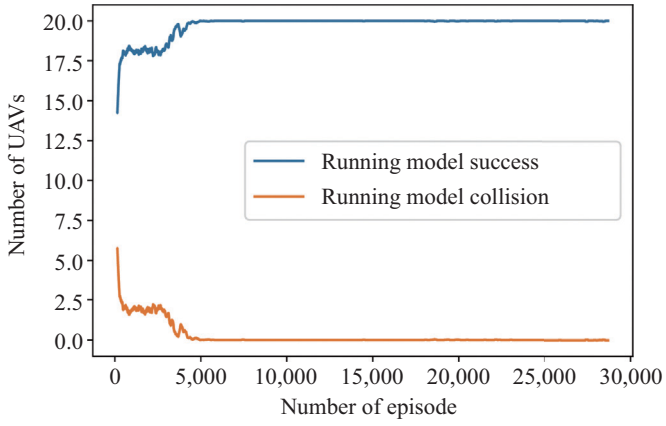



**Fig. 23.** MARL model training with 20 vehicles.

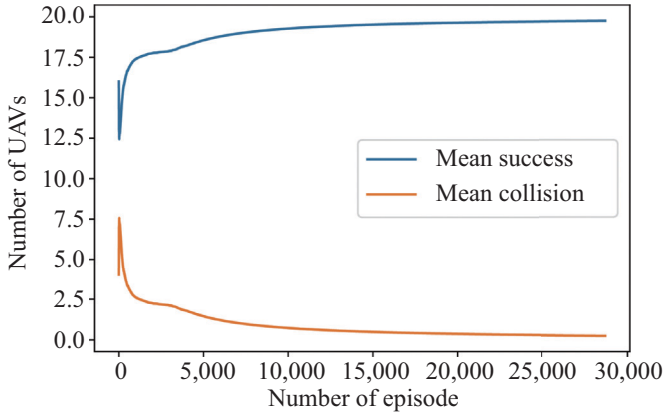



**Fig. 24.** MARL model performance with 20 vehicles.

our model demonstrates better learning performance and generates improved policies for ensuring safe separations at the intersection.

The density of air traffic significantly impacts the performance of our MARL approach, particularly in the intersection scenario. A higher density implies a greater number of aircraft in the airspace simultaneously, increasing the likelihood of conflicts and safety issues during the intersection crossing. In the BlueSky simulation environment, increasing the aircraft density introduces more complex air traffic scenarios, requiring real-time management of a larger number of aircraft. This poses a greater challenge for ensuring safe separation between the aircraft, ultimately putting the reinforcement learning-based autonomous air traffic management system to the test.

To assess the influence of aircraft density on our MARL approach's performance in the intersection case, we conduct tests with 20 aircraft and record the model's behavior, as shown in Fig. 23. The results indicate that our model successfully resolves conflicts at the intersection, even with a high number of vehicles present. The proposed approach efficiently handles increased aircraft density, and its performance remains robust even as the density increases. In fact, Fig. 24 shows that training and testing our model with a higher number of

vehicles will be giving slightly more accurate results for AAM operations, as has been already observed in the merging scenario. The higher number of aircraft in the model allows the algorithm to encounter and learn from a wider range of scenarios and situations, which can improve its ability to handle the conflicts at the intersection. As the number of aircraft increases, the model has more opportunities to experience different scenarios, which can help it learn to identify patterns and strategies that are more effective when dealing with larger numbers of aircraft. Additionally, the increased complexity of the scenario with more aircraft can help the model develop more robust policies that can handle a wider range of situations, leading to better overall performance. Therefore, training and testing the model with a higher number of aircraft can lead to more accurate and effective results for autonomous air traffic management operations. Given that the traffic scenarios simulated in this case assume the same flight level, the maximum number of vehicles in the same corridors is limited by safety conditions dictated by the AAM concept of operation. Consequently, further testing with higher aircraft densities is necessary to investigate the performance and potential limitations of our model comprehensively.

The presented simulation results demonstrate the effectiveness of our proposed data-driven air traffic management system in handling complex air traffic scenarios and ensuring safe separation between aircraft. The merging and intersection case scenarios illustrate the capability of our MARL model to address critical challenges in AAM operations. However, it is important to continue testing and refining the system to enhance its reliability, especially for practical real-world implementations. Additionally, considering other factors such as the geometry of merging points, vehicle size and speed, and external influences (e.g., weather conditions and AAM system constraints) will contribute to a more comprehensive evaluation of the algorithm's performance in diverse conditions.

## 6. Conclusion

Autonomous traffic control and management at merging points and intersections are raising problems in advanced air mobility (AAM). In conclusion, this paper addresses the challenge of autonomous traffic control and management at merging points and intersections in advanced air mobility (AAM). The proposed solution involves formulating and solving model-free problems for AAM vehicles operating within air corridors, with the aim of achieving safer and more efficient AAM missions. A multi-agent reinforcement learning (MARL) model is developed to tackle the problem, and a neural network architecture is designed to train and evaluate the model using the BlueSky environment. The use of LSTM architecture enables effective encoding of intruder information and facilitates decision-making. The trained model provides policies for each agent to follow during flight, ensuring conflict-free operations at merging points and intersections. The simulation results demonstrate the convergence of the proposed approach, highlighting the successful guidance of vehicles in different scenarios. This research contributes to the advancement of autonomous air traffic control in the context of AAM, paving the way for enhanced safety and efficiency in future air transportation systems.

## CRediT authorship contribution statement

Sabrullah Deniz: Methodology, Validation, Formal analysis, Writing - Original Draft.

Yufei Wu: Writing - Review & Editing.

Yang Shi: Writing - Review & Editing.

Zhenbo Wang: Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

## Data availability statement

The data and materials used to support the findings of this study are available from the corresponding author upon reasonable request.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was funded in part by the National Science Foundation (NSF) CAREER Award CMMI-2237215.

## References

[1] Goodrich KH, Theodore CR. Description of the nasa urban air mobility maturity level (uml) scale. In: AIAA Scitech 2021 forum; 2021. p. 1627.
[2] Hasan S. Urban air mobility (uam) market study. Tech Rep 2019.
[3] Holden J, Goel N. Fast-forwarding to a future of on-demand urban air transportation. 2016. San Francisco, CA.
[4] Airbus. Urban air mobility by airbus. 2018.
[5] Corgan. Connect evolved. 2019.
[6] Forecast FA. Office of aviation policy and plans (apo-100) faa u.s. passenger airline forecasts, fiscal years 2020-2040. 2020.
[7] FAA. Concept of operations v2.0", enabling civ. low-altitude airsp. unmanned aircr. syst. oper. 2020. https://utm.arc.nasa.gov/index.shtml.
[8] Bradford S. Urban air mobility (uam) concept of operations v1. 0. Boston: EmbraerX; 2020. p. 5.
[9] Johnson M, Jung J, Rios J, Mercer J, Homola J, Prevot T, et al. Flight test evaluation of an unmanned aircraft system traffic management (utm) concept for multiple beyond-visual-line-of-sight operations. In: USA/Europe air traffic management research and development Seminar (ATM2017). ARC-E-DAA-TN39084; 2017.
[10] Jung J, Rios JL, Drew CR, Modi HC, Jobe KK. Small unmanned aircraft system off-nominal operations reporting system. 2020.
[11] Google. Alphago — deepmind. 2018.
[12] OpenAI. Openai. 2019.
[13] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8): 1735–80.
[14] Hoekstra JM, Ellerbroek J. Bluesky atc simulator project: an open data and open source approach. In: Proceedings of the 7th international conference on research in air transportation, vol. 131. FAA/Eurocontrol USA/Europe; 2016. p. 132.
[15] Bouton M, Nakhaei A, Fujimura K, Kochenderfer MJ. Cooperation-aware reinforcement learning for merging in dense traffic. In: 2019 IEEE intelligent transportation systems conference (ITSC). IEEE; 2019. p. 3441–7.
[16] Liang X, Du X, Wang G, Han Z. Deep reinforcement learning for traffic light control in vehicular networks. arXiv preprint arXiv:1803.11115 2018.
[17] Genders W, Razavi S. Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142 2016.
[18] Chen D, Li Z, Wang Y, Jiang L, Wang Y. Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic. arXiv preprint arXiv:2105.05701 2021.
[19] Erzberger H. Automated conflict resolution for air traffic control (25Th Int. Congr. Aeronaut. Sci., no. March. 2014. p. 1–28.
[20] Erzberger H, Heere K. Algorithm and operational concept for resolving short-range conflicts. Proc Inst Mech Eng G J Aerosp Eng 2010;224(2):225–43.
[21] Tumer K, Agogino AK. Adaptive management of air traffic flow: a multiagent coordination approach.. In: AAAI; 2008. p. 1581–4.
[22] Brittain M, Yang X, Wei P. A deep multi-agent reinforcement learning approach to autonomous separation assurance. arXiv preprint arXiv:2003.08353 2020.
[23] Chu T, Wang J, Codecà L, Li Z. Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans Intell Transport Syst 2019;21(3):1086–95.
[24] Schuchardt BI, Geister D, Lüken T, Knabe F, Metz IC, Peinecke N, et al. Air traffic management as a vital part of urban air mobility—a review of dlr's research work from 1995 to 2022. Aerospace 2023;10(1):81.
[25] Pinto Neto EC, Baum DM, Almeida Jr JRd, Camargo Jr JB, Cugnasca PS. Deep learning in air traffic management (atm): a survey on applications, opportunities, and open challenges. Aerospace 2023;10(4):358.
[26] de Oliveira ÍR, Neto ECP, Matsumoto TT, Yu H. Decentralized air traffic management for advanced air mobility. In: 2021 integrated communications navigation and surveillance conference (ICNS). IEEE; 2021. p. 1–8.
[27] Deniz S, Wang Z. A multi-agent reinforcement learning approach to traffic control at future urban air mobility intersections. In: AIAA SCITECH 2022 Forum; 2022. p. 1509.
[28] Deniz S, Wu Y, Shi Y, Wang Z. A multi-agent reinforcement learning approach to traffic control at merging point of urban air mobility. In: AIAA AVIATION 2022 forum; 2022. p. 3912.
[29] Elevate U. Operations inside corridors. 2020. October.
[30] Morales EF, Zaragoza JH. An introduction to reinforcement learning. In: Decision theory models for applications in artificial intelligence: concepts and solutions. IGI Global; 2012. p. 63–80.
[31] Garcia F, Rachelson E. Markov decision processes, Markov decision processes in artificial intelligence. 2013. p. 1–38.
[32] Pham D-T, Tran NP, Goh SK, Alam S, Duong V. Reinforcement learning for two-aircraft conflict resolution in the presence of uncertainty. In: 2019 IEEE-RIVF international conference on computing and communication technologies (RIVF). IEEE; 2019. p. 1–6.

[33] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 2017.
[34] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016. p. 1928–37.
[35] Nachum O, Norouzi M, Xu K, Schuurmans D. Bridging the gap between value and policy based reinforcement learning. Adv Neural Inf Process Syst 2017;30.
[36] Peters J. Policy gradient methods. Scholarpedia 2010;5(11):3698.
[37] Sutton RS, Barto AG. Reinforcement learning: an introduction. MIT press; 2018.
[38] Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. Adv Neural Inf Process Syst 1999;12.
[39] Morimura T, Uchibe E, Doya K. Natural actor-critic with baseline adjustment for variance reduction. Artif Life Robot 2008;13(1):275–9.
[40] Berner C, Brockman G, Chan B, Cheung V, Dębiak P, Dennison C, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680 2019.
[41] Busoniu L, Babuska R, De Schutter B. A comprehensive survey of multiagent reinforcement learning. IEEE Trans Sys Man Cybernet Part C (Applications and Reviews) 2008;38(2):156–72.
[42] Kumar RR, Varakantham P. On solving cooperative marl problems with a few good experiences. arXiv preprint arXiv:2001.07993 2020.
[43] Tan M. Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning; 1993. p. 330–7.
[44] Matignon L, Laurent GJ, Le Fort-Piat N. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. Knowl Eng Rev 2012;27(1):1–31.
[45] Kraemer L, Banerjee B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing 2016;190:82–94.
[46] Weiss G. Multiagent systems: a modern approach to distributed artificial intelligence. Int J Comput Intell Appl 2001;1:331–4.
[47] Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 2015.
[48] Wang Y, He H, Tan X. Truly proximal policy optimization. In: Uncertainty in artificial intelligence. PMLR; 2020. p. 113–22.
[49] Chen G. A new framework for multi-agent reinforcement learning–centralized training and exploration with decentralized execution via policy distillation. arXiv preprint arXiv:1910.09152 2019.
[50] Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation functions: comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378 2018.
[51] Niu Z, Zhong G, Yu H. A review on the attention mechanism of deep learning. Neurocomputing 2021;452:48–62.
[52] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13(2).
[53] Probst P, Wright MN, Boulesteix A-L. Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Rev: Data Min Knowl Discov 2019;9(3):e1301.
[54] Wu J, Chen X-Y, Zhang H, Xiong L-D, Lei H, Deng S-H. Hyperparameter optimization for machine learning models based on bayesian optimization. J Electr Sci Tech 2019;17(1):26–40.