# Comparing attention-based methods with long short-term memory for state encoding in reinforcement learning-based separation management

Groot, D. J.; Ellerbroek, J.; Hoekstra, J. M.

**DOI**
10.1016/j.engappai.2025.111592

**Publication date**
2025

**Document Version**
Final published version

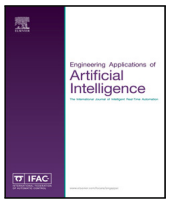
**Published in**
Engineering Applications of Artificial Intelligence

Engineering Applications of Artificial Intelligence 159 (2025) 111592

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research paper

# Comparing attention-based methods with long short-term memory for state encoding in reinforcement learning-based separation management

D.J. Groot *, J. Ellerbroek, J.M. Hoekstra

*Delft University of Technology, Kluyverweg 1, 2629 HS, Delft, Zuid-Holland, The Netherlands*

ARTICLE INFO

*Keywords:*
Deep reinforcement learning
Soft Actor–Critic
Long short-term memory
Attention methods
Separation managements
Air traffic control

ABSTRACT

Reinforcement learning (RL) is a method that has been studied extensively for the task of conflict-resolution and separation management within air traffic control, offering advantages over analytical methods. One key challenge associated with RL for this task is the construction of the input vector. Because the number of agents in the airspace varies, methods that can handle dynamic number of agents are required. Various methods exist, for example, selecting a fixed number of aircraft, or using methods such as recurrent neural networks or attention to encode the information. Multiple studies have shown promising results using these encoder methods, however, studies comparing these methods are limited and the results remain inconclusive on which method works better. To address this issue, this paper compares different input encoding methods: three different attention methods – scaled dot-product, additive and context aware attention – and long short-term memory (LSTM) with three different sorting strategies. These methods are used as input encoders for different models trained with the Soft Actor–Critic algorithm for separation management in high traffic density scenarios. It is found that additive attention is the most effective at increasing the total safety and maximizing path efficiency, outperforming the commonly used scaled dot-product attention and LSTM. Additionally, it is shown that the order of the input sequence significantly impacts the performance of the LSTM based input encoder. This is in contrast with the attention methods, which are sequence-independent and therefore do not suffer from biases introduced by the order of the input sequence.

## 1. Introduction

The increasing scale and complexity of civilian air traffic have placed growing demands on modern air traffic control/management (ATC/ATM) systems. As global aviation continues to expand, managing busy airspaces efficiently and safely has become more challenging, especially with the current, largely manual, ATC procedures. To address these challenges and reduce the workload of human controllers, there is a strong incentive to investigate higher levels of automation for ATC operations, as mentioned in the European ATM Master Plan (Undertaking et al., 2016). One promising method for achieving these higher levels of automation is the use of (Multi-Agent) Reinforcement Learning (RL/MARL), a branch of machine learning that focuses on sequential decision-making problems, with models learning to map states to actions in order to maximize an accumulated reward (Barto, 2021).

RL has demonstrated to be successful in various sequential decision-making domains, such as video games, robotics, and even for training large language models (LLMs) like Generative Pre-Trained Transformers (GPTs), improving their interactions with humans (Achiam et al., 2023). Within the field of ATC, RL has shown a lot of promising results within the area of Conflict Resolution (CR) and separation management, which is the process of ensuring safe distances between different aircraft (Wang et al., 2022; Razzaghi et al., 2024)

Compared to many traditional RL applications, separation management in ATC has a very specific characteristic: the number of aircraft within a given airspace is not likely to be constant, resulting in a size-varying state space. This poses a problem for conventional RL algorithms that rely on neural networks expecting inputs of fixed size. This requires alterations to the strategies used for constructing the input vector.

One approach is to focus solely on one-to-one conflicts, reducing the problem to a more isolated, simple representation. The learned policies can then be applied to aircraft when they are getting too close, ignoring surrounding aircraft (Pham et al., 2019). Other studies have instead focused on keeping the number of aircraft in the airspace constant. Limiting the study to smaller numbers of aircraft to ensure that the state space does not become too large (Wang et al., 2019; Wen et al., 2019). This however, severely limits the scalability of the trained model

* Corresponding author.
*E-mail addresses:* d.j.groot@tudelft.nl (D.J. Groot), j.ellerbroek@tudelft.nl (J. Ellerbroek), j.m.hoekstra@tudelft.nl (J.M. Hoekstra).

https://doi.org/10.1016/j.engappai.2025.111592
Received 18 November 2024; Received in revised form 15 May 2025; Accepted 19 June 2025
Available online 5 July 2025

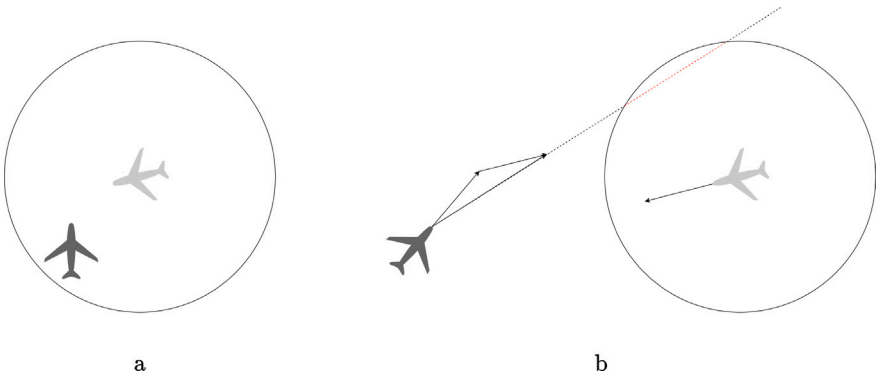



**Fig. 1.** (a.) illustration of an intrusion, when two aircraft violate the required separation minima. (b.) illustration of a conflict, when the relative velocity of two aircraft predicts an intrusion in the future.

to environments where the number of aircraft does not match that of the training scenarios.

A slightly more flexible approach is to select a fixed number of aircraft to be included in the state vector, based on some selection criteria, such as distance or time to closest point of approach. Many studies have shown that this strategy is effective (Ribeiro et al., 2022a; Groot et al., 2024a; Li et al., 2019). However, a downside presents itself in the determination of the importance of the aircraft, which can be non-trivial in certain scenarios such as parallel or head-on conflicts. Additionally, this strategy formally changes the problem to a Partially Observable MDP (POMDP), and requires an additional method for constructing the state vector when the total number of aircraft in the airspace is lower than the fixed number used for training.

An alternative approach for constructing the input state vector that has shown promising results, without introducing partial observability, is using Recurrent Neural Networks (RNNs) (Brittain and Wei, 2021; Dalmau and Allard, 2020). RNNs process the agents (i.e. aircraft) in the environment sequentially, aggregating the information into a hidden state. This hidden state is then used as the input vector for the RL model. However, using RNNs for this purpose has a potential limitation: they are heavily dependent on the order of the input sequence. Because of this, the output of the RNN network, and thus the input vector to the model, differs depending on the order in which the aircraft in the environment are presented to the RNN network. This sequence dependence can result in the forgetting of earlier elements in longer sequences, introducing biases based on the method used for ordering the sequence and changes performance when the order of the input sequence is altered. While this is a known shortcoming, there are to date no studies that have quantified this impact. Moreover, because RNNs sequentially loop through the data, they cannot effectively use hardware optimized for parallel processing.

To address the last issue, Vaswani et al. (2017), investigated if a technique called attention can be used as an alternative to RNNs for processing sequential data, specifically for natural language processing (NLP) purposes. This attention-based network architecture, called the Transformer, not only significantly decreased training time when compared with RNNs, but also outperformed these RNN-based models on various benchmarks. Additionally, these methods that focus solely on attention are naturally sequence-independent, and normally include additional sequence encoding layers. This property can be advantageous when using attention for input encoding in MARL applications, as it does not generate the biases introduced by the order of the input sequence present in RNNs.

Because of these advantages, several studies have investigated attention-based methods for input encoding in RL for ATC applications (Brittain et al., 2020, 2024; Deniz et al., 2024; Groot et al., 2023). These studies have primarily looked at one specific type of attention; scaled dot-product attention. While this method obtains state-of-the-art performance on NLP tasks, other attention methods such as additive attention do exist (Bahdanau et al., 2014). Considering the large domain differences between input encoding for separation assurance with RL and NLP, it warrants the investigation between these methods for this domain.

In addition, few studies have directly compared attention with RNN-based architectures for this task. For example, Brittain et al. found that scaled dot-product attention outperformed long short-term memory (LSTM) architectures (Brittain et al., 2020), while Deniz et al. reported the opposite, with LSTM obtaining better performance (Deniz et al., 2024). Furthermore, both studies were limited to simple scenarios with fixed airway structures and a discrete action space limited to speed changes. As a result, the total interaction between all aircraft in the airspace is limited, making the encoding of 'important' aircraft more trivial. As such, these scenarios might not fully reflect the maximum requirements of a high traffic density environment with mixed interactions, leaving the question which input encoding strategy is more effective partially unanswered.

The aim of this study is therefore to answer two open questions for reinforcement learning applications in ATC.

1. How does the performance of attention based methods compare with LSTM for the task of input encoding?
2. How much does the input sequence order for LSTM influence the final performance of the model?

The remainder of this paper will first introduce the problem of conflict resolution and its associated definitions. Next, it will detail the methods employed in this study: the Soft Actor–Critic reinforcement learning algorithm, the Long Short-Term Memory RNN, and various attention mechanisms. This is followed by a section on the experimental setup and a combined results and discussion section, highlighting the differences between the methods.

## 2. Aircraft conflict resolution and separation management

The main focus of this study is on the efficacy of different input state encoders at facilitating the task of tactical separation management in ATC through RL. To provide the necessary background information for understanding the remainder of this paper, this section will explain the concepts of intrusions and conflicts.

### 2.1. Intrusions

An intrusion, also called Loss-of-Separation, is an event in which two aircraft violate predefined separation minima, and is considered a serious breach of safety. These separation minima are defined based on uncertainties in the positions (state) of the aircraft. It is therefore essential that the number of intrusions is minimized to limit the chance of a collision. An example of an intrusion is shown in Fig. 1a. For the purpose of this study a separation minimum of 5NM is used, which is commonly used in en-route airspace.

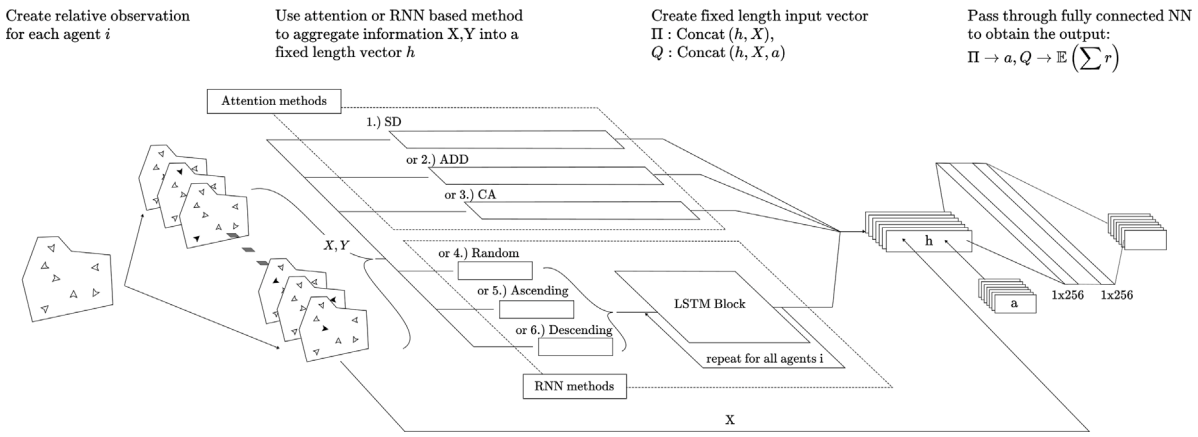



**Fig. 2.** Overview of the six network architectures used to construct the input vector $h$ from the environment's state. The methods are: (1) Scaled Dot-Product Attention, (2) Additive Attention, (3) Context-Aware Attention, (4) Randomly Sorted LSTM, (5) Ascending Distance LSTM, (6) Descending Distance LSTM.

### 2.2. Conflicts

In this study, the term conflict refers to a predicted intrusion based on a linear extrapolation of the current state of the aircraft (state-based conflict detection). An example of a conflict is provided in Fig. 1b.

### 2.3. Tactical separation management

Once a conflict is detected it is important that this is resolved to ensure that no intrusion happens. Analytical methods, such as the modified voltage potential (MVP) algorithm, determine the shortest way out of the conflict for aircraft currently in conflict within a given time-window (Hoekstra et al., 2002). With reinforcement learning, however, it is possible to have the model learn when to act, dynamically changing the length of this time-window (Ribeiro et al., 2022b) or even acting when no conflict is detected to increase safety margins (Groot et al., 2024a). This gives reinforcement learning an advantage over analytical methods under certain conditions. In this study the approach used is similar as that in Groot et al. (2024a), which allows the agents to deviate from their nominal path even when not in conflict.

## 3. Methodology

### 3.1. Soft Actor-Critic

For this study the Soft Actor–Critic (SAC) RL algorithm (Haarnoja et al., 2018) is used to train the models, as it has been shown to be effective at the task of CR with RL and has been combined with attention networks before (Groot et al., 2023). SAC is an off-policy RL algorithm that trains two different types of networks, a stochastic actor network (policy $\Pi$) that is used for mapping states to actions, and the critic networks, $Q$. These critic networks estimate the expected sum of rewards, also called the return, for the current state, action, and policy. This expected return is then used to update the policy in the direction of higher returns. For the purpose of this study the actor and critic networks all use a fully connected neural network of 2 layers and 256 neurons and the actor network is shared for all agents in the environment. To construct the input vector, h, for these neural networks, six different methods have been implemented: three attention-based methods (Section 3.2) and three LSTM-based methods (Section 3.3). Additionally, the actor network only uses the state of the environment to construct the input. The critic on the other hand also uses the agent's actions in the input. A summary of the total network architectures used is given in Fig. 2.

The hyperparameters used are given in Table 1. Notable changes to the original hyperparameters of Haarnoja et al. are the larger batch size of 2048 and discount factor $\gamma$ of 0.99, which were found to increase the stability of training, resulting in higher asymptotic performance of the models.

**Table 1**
Hyperparameters used for the SAC algorithm.

| Parameter | Value |
|---|---|
| Optimizer | Adam |
| Alpha Learnrate | 3e−4 |
| Actor Learnrate | 3e−4 |
| Critic Learnrate | 3e−3 |
| Discount factor ($\gamma$) | 0.99 |
| Memory buffer size | 20e6 |
| Sample size | 2048 |
| Smoothing coefficient ($\tau$) | 5e−3 |
| Network update frequency | 8 |

### 3.2. Attention mechanisms

Attention mechanisms were first introduced by Bahdanau et al. to enhance the performance of Recurrent Neural Networks (RNNs) when processing longer sequences (Bahdanau et al., 2014). These mechanisms work by determining attention scores for all elements in the sequence by comparing these elements. The input sequence is then weighted with these attention scores before being used as an input for the RNN, ensuring that more important elements are given higher weights, which in turn increases their contribution to the output vector of the RNN.

In 2017, Vaswani et al. proposed to use attention mechanisms as a stand-alone method instead, leading to significant reductions in training costs while also improving on the state of the art in machine translation tasks (Vaswani et al., 2017). This work also introduced an attention mechanism known as scaled dot-product attention, which demonstrated comparable performance to additive attention in machine translation tasks, but with lower computational requirements. However, unlike machine translation, MARL often involves physical state representations in the input. As an example consider two aircraft flying on orthogonal paths. In this case the two aircraft can be on a collision course, depending on the direction of the velocity and the current aircraft positional states. Because the dot-product between two orthogonal vectors is zero, using a naive (unweighted) dot-product attention method for these two vectors would result in an attention score of zero. In contrast, additive attention would result in an attention score based on the relative velocity of these aircraft, which carries more physical information. It is therefore hypothesized that additive attention may outperform scaled dot-product attention when applied to input encoding for MARL-based conflict resolution.

In both of these attention mechanisms, the attention weights are computed using a Query $Q$ and Key $K$. These weights are then multiplied with the Value $V$ to determine the contribution of the input to the hidden state $h$. When these attention methods are used to

encode the environment from the perspective of a single agent (the ownship, i.e., the aircraft making the decision), $Q$ is calculated using the ownship's observation $X$, while $K$ and $V$ are calculated using the relative observations of the other agents, $Y$. However, since the relation between $Q$ and $K$ is reduced to a scalar attention weight, more complex relations between $X$ and $Y$ may be lost. To identify whether this reduction affects performance, this paper proposes an additional attention method specific for MARL-based separation management: context aware attention. Context aware attention is based on additive attention, but computes $V$ using both $X$ and $Y$, potentially allowing $V$ to capture the relationship between the ownship and other agents more effectively.

In summary, this paper evaluates three different attention mechanisms: scaled dot-product (SD) attention (Vaswani et al., 2017), additive (ADD) attention (Bahdanau et al., 2014) and the proposed context aware (CA) attention. All of these methods adhere to the multi-head attention network structure introduced by Vaswani et al. with minor variations. The remainder of this section details the implementation and specific differences among these methods.

First the Query, $Q_i$, Key, $K_i$, and Value, $V_i$, matrices for each head 'i' are calculated using the ownship's observation $X$ and relative observations $Y$, and their respective (trainable) weight matrices $W_{Qi}$, $W_{Vi}$, $W_{Ki}$ according to Eq. (1). One exception for the calculation of $V_i$ is for the CA attention method, which aims to capture the relation between $X$ and $Y$ in $V$ using Eq. (2).

$$Q_i = XW_{Q_i}, \quad K_i = YW_{K_i}, \quad V_i = YW_{V_i} \tag{1}$$

$$V_i = \text{Concat}(X, Y)W_{V_i} \tag{2}$$

These matrices are then used to determine the per-head attention vector. For the SD attention Eq. (3) is used, whereas for ADD and CA attention, Eq. (4) is used.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{3}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\tanh(Q + K + b)\right)V \tag{4}$$

Finally the different heads are concatenated to create the entire multi-head attention vector, $h$, as per Eqs. (5) and (6). For each of these heads, a different set of weights, $W$, is used, allowing each head to learn to focus on different things. For all experiments in this paper, a total of three heads are used, where each head is the same size as $Y$.

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i) \tag{5}$$

$$h = \text{Concat}(\text{head}_1, \text{head}_2, \text{head}_3) \tag{6}$$

This resulting vector, $h$, then serves as the fixed length input vector encoding the state of the other agents in the environment for the fully connected neural network described in Section 3.1.

### 3.3. LSTM

The Long Short-Term Memory (LSTM) network is a specific type of RNN that was developed to better deal with long term dependencies in a sequence (Hochreiter, 1997). An LSTM network does this by adding an additional cell state, $c_t$, on top of the conventional hidden/output state. This cell state can store information from earlier in the sequence through forget and input gates. An illustration of such an LSTM block is given in Fig. 3. The remainder of this section will elaborate on the specific implementation of LSTM for this paper and the corresponding equations.

Because LSTM networks process inputs sequentially, the order of the input sequence influences the network's output, introducing biases that can negatively impact model performance. To evaluate the extent of this effect, three different sorting strategies for the input sequence Y, which contains the relative states of the other agents in the environment, are used, each representing a different assumption about the decision relevance of nearby aircraft. Here, decision relevance refers to the likelihood that a given aircraft will influence the agent's decision-making, for example, nearby aircraft are more likely to be relevant than distant ones.

- **Random LSTM** - For the Random LSTM method, the input sequence $Y$ is randomly shuffled. Ensuring that the decision relevance of a drawn sample $y$ is not correlated with its position in the sequence.
- **Ascending LSTM** - For the Ascending LSTM method the input sequence is sorted from furthest away to closest, aiming to create a positive correlation between the decision relevance and position in the sequence, placing more relevant aircraft later in the sequence.
- **Descending LSTM** - For the Descending LSTM method the input sequence is sorted from closest aircraft to furthest. This places less relevant aircraft later in the sequence.

The implementation of the LSTM is based on the Pytorch implementation (Paszke et al., 2019) and follows the update rules for determining the new cell and hidden state in accordance with the schematics illustrated in Fig. 3. First the forget gate, $f_t$, and input gate, $i_t$, are calculated using Eqs. (7) and (8) respectively. Here $\sigma$ is the sigmoid activation function, $h_{t-1}$ is the resulting hidden state from the previous element in the sequence and is a vector of zeros for $t = 0$, and finally $y_t \in Y$ is the current input to the LSTM drawn from the sorted input sequence $Y$. The $W$s and $b$s are trainable weights and biases respectively. The resulting forget and input gates are vectors equal to the size of the cell state, and are strictly between 0 and 1 through the sigmoid function. This allows the network to selectively remove and add elements to the cell state.

$$f_t = \sigma(W_{fh} \cdot h_{t-1} + W_{fy} \cdot y_t + b_f) \tag{7}$$

$$i_t = \sigma(W_{ih} \cdot h_{t-1} + W_{iy} \cdot y_t + b_i) \tag{8}$$

Then, a candidate cell state, $\tilde{c}_t$ is generated using Eq. (9), which represents the information from the previous hidden state, $h_{t-1}$, and input state, $y_t$, should be added to the new cell state $c_t$. The information from the previous cell state, $c_{t-1}$ and the candidate cell state are then multiplied with the forget and input gate, and summed according to Eq. (10) to obtain the new cell state.

$$\tilde{c}_t = \tanh(W_{ch} \cdot h_{t-1} + W_{cy}y_t + b_c) \tag{9}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{10}$$

Finally, $x_t$, $h_{t-1}$, and $c_t$ are combined using Eq. (11) to obtain the new hidden state, $h_t$. This process is then repeated until all inputs states, $y_t \in Y$, have been passed through the network, after which the most recent hidden state, $h$, is used as the output for the network.

$$h_t = \sigma(W_{oh} \cdot h_{t-1} + W_{oy} \cdot y_t + b_o) \cdot \tanh(c_t) \tag{11}$$

To ensure a fair comparison between both methods, both the hidden, and cell state of the LSTM are of size $3 \cdot \|Y\|$, matching the size of the $h$ vector resulting from three heads for the attention methods.

## 4. Simulation experiment

For training and evaluating of the models corresponding the different methods introduced in Section 3, randomly generated air traffic scenarios are used. This section first outlines the rules used for the random creation of these scenarios, followed by a description of the ATCenv simulator used in the experiments. Next, the Markov Decision Process (MDP) formalization of the MARL task is given. Finally, the dependent variables used for evaluating the models performance are explained.

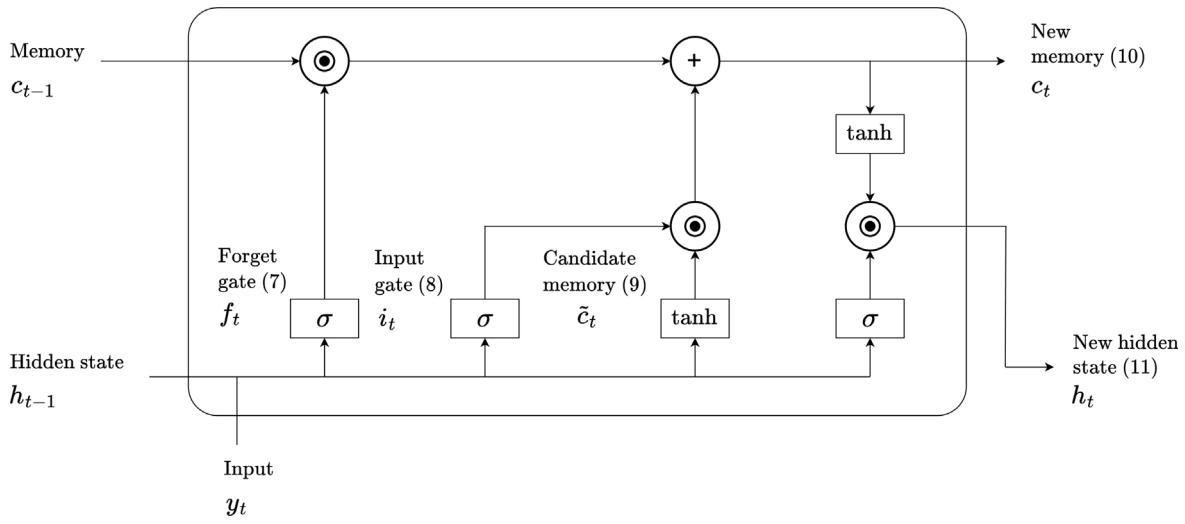



**Fig. 3.** Update rules for the LSTM block. Equations in text corresponding to the operations in the illustration are given between brackets.

### 4.1. Simulation scenario

The simulation scenarios are designed to test the models' efficacy for conflict resolution and intrusion free multi-agent navigation. To do this random polygons of size $10\,000\,\text{km}^2 \pm 10\%$ are generated to serve as the airspace sector. This airspace sector is then filled with 20 spatially separated aircraft,[1] each with a corresponding destination and their heading initialized such that it faces in the direction of their destination. Because both the shape of the airspace, and the initial conditions of the aircraft are randomly generated, the models encounter a large diversity of traffic scenarios during training, and never train on the same scenario twice. The final combination of sector size and number of aircraft leads to higher traffic densities than normally expected in the airspace. This higher traffic density is beneficial for the training of the models, as it has been shown that the efficacy of RL method training increases at higher traffic densities (Groot et al., 2024a). This effect can be attributed to the fact that at lower traffic densities intrusion events are sparse, resulting in many update steps to the network that do not correspond to intrusion events. Higher traffic densities also increase the difficulty of the task, better highlighting differences between the different methods. Fig. 4 shows an example initial state following these rules. In this figure, the circles correspond to half the radius of minimum separation, meaning that if two circles overlap there is an intrusion.

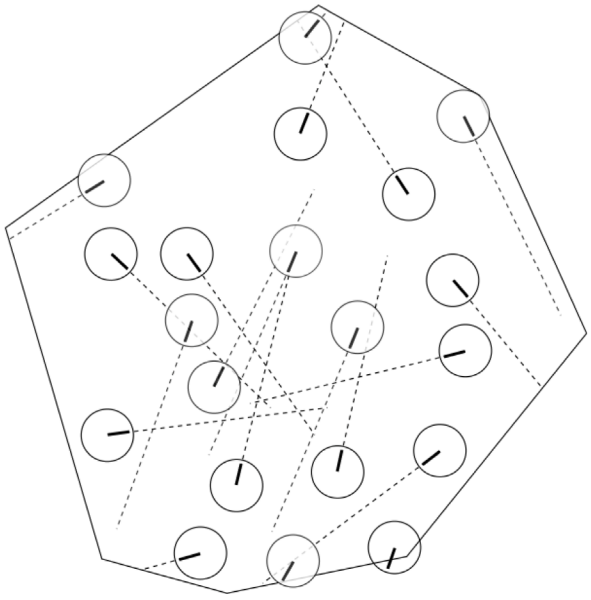


**Fig. 4.** Example initial conditions for the traffic scenarios. Circles represent half of the separation requirements, e.g. when the circles overlap there is an intrusions.

### 4.2. Simulator: ATCenv

The simulator used to run the scenarios is based on ATCenv.[2] ATCenv was originally created by Eurocontrol for a reinforcement learning competition for separation management, and does not consider aircraft dynamics. Instead, the states are linearly propagated in time with a time-step of 5 s and any changes to the states are applied directly. This allows models to be evaluated solely on the quality of their paths and the interaction between the different agents, similar to the Multi Particle Environments commonly used for MARL studies (Lowe et al., 2017), without also having to learn underlying aircraft dynamics.

### 4.3. Markov decision process formulation

In this section, the Markov Decision Process (MDP) used to model the MARL task within the air traffic control scenarios is defined. Any MDP is characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition probabilities, and $\mathcal{R}$ the reward function. For the purpose of this study $\mathcal{P}$ is fully encompassed in the used simulator, ATCenv, and therefore not further elaborated upon. The remainder of this section detail the other components of the MDP, including the observation vectors X and Y representing $\mathcal{S}$.

#### 4.3.1. Observation vectors X and Y

The observation vectors $X$ and $Y$ are used to represent the environments state $\mathcal{S}$ from the perspective of each agent, as it was shown that using representations of the environment relative to each agent improves learning speed and overall performance of the trained models (Groot et al., 2023). Here $X$ represents the ownship state, and $Y$ represents the agents state relative to the ownship. The contents of

[1] Note that this number of agents is constant due to the requirements related to the memory buffer and matrix operations used during batched training of the networks. The methods, however, work with any number of agents after training, when the memory buffer and batch operations are no longer required.

[2] Source code for the original ATCenv is available at: "https://github.com/ramondalmau/atcenv". The version used for this paper can be found at: "https://github.com/jangroter/atcenv/tree/attention_study".

**Table 2**
Features included in the observation vectors $X$ and $Y$.

| | Variable | Description |
|---|---|---|
| X | $x_i$ | x position |
| | $y_i$ | y position |
| | $vx_i$ | Speed in the x direction |
| | $vy_i$ | Speed in the y direction |
| | $v_i$ | Total aircraft speed |
| | $\cos(\delta_i)$ | Cosine of the track deviation $\delta$ |
| | $\sin(\delta_i)$ | Sine of the track deviation $\delta$ |
| Y | $x_{i,j}$ | Relative x position |
| | $y_{i,j}$ | Relative y position |
| | $vx_{i,j}$ | Relative speed in the x direction |
| | $vy_{i,j}$ | Relative speed in the y direction |
| | $d_{i,j}$ | Distance between the aircraft |

**Table 3**
Action bounds for the different available actions to the RL methods.

| Action | Bounds |
|---|---|
| Heading | [−25°, +25°] |
| Speed | [−50 m/s, +50 m/s] bounded by [200 m/s, 250 m/s] |

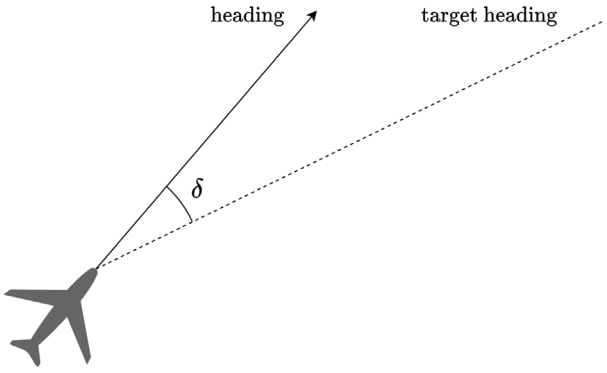



**Fig. 5.** Illustration of the drift angle $\delta$ for a single agent.

these vectors are given in Table 2. Additionally, all the values in these vectors are normalized to help with stability during the initial phases of training.

#### 4.3.2. Action space

The action space $\mathcal{A}$ of the environment is continuous and allows for 2 types of actions per agent per time-step, a heading change and a speed change. The bounds of these actions are given in Table 3. It is noted that the values given in this table are not realistic considering actual aircraft dynamics, however, they are used to allow for more flexibility in the paths generated by the agents, resulting in the performance only being evaluated on the quality of the paths, and not the understanding of the environment dynamics.

#### 4.3.3. Reward function

The reward function $\mathcal{R}$ is built up out of two components, and is strictly negative. The total reward function can be seen in Eq. (12). The first part of the equation penalizes the drift, $\delta$, of the agent with respect to its destination. This drift is also illustrated in Fig. 5. The second part of the equation is a simple boolean operation which equals 0 when the aircraft is intrusion-free and −1 if the aircraft is not. The magnitude of the weight $w_{drift}$ is set at −0.1 and the drift angle $\delta$ is represented in radians. This ratio was determined through empirical tests, focusing on a trade-off between safety and efficiency, and is identical to that used in previous research (Groot et al., 2023).

$$\mathcal{R} = w_{drift} \cdot |\delta| + \begin{cases} -1 & \text{if intrusion} \\ 0 & \text{if not intrusion} \end{cases} \tag{12}$$

### 4.4. Dependent variables

The dependent variables are the measures used to evaluate the performance of the different methods and are based on the reward, number of intrusions, average track deviation and required computation time of the different methods.

#### 4.4.1. Total reward

The total reward, or return, shows the average reward obtained by each of the agents over a single episode. Because the reward is used to train the critic, and hence the policy, it directly relates to the performance of the method and is considered the most important parameter when discussing the performance of the different methods.

#### 4.4.2. Total number of intrusions

Given that the reward function in this study comprises two components, track deviation and intrusions, different methods may achieve similar overall rewards through different policies. Therefore, it is important to evaluate the individual components contributing to the reward.

The total number of intrusions directly relates to the safety of the learned policy. Since effective CR relies on an accurate representation of the surrounding state, the intrusion metric also serves as an indicator of the effectiveness of the input state encoder method used.

#### 4.4.3. Average track deviation

Average track deviation is used as a measure for path efficiency of the different policies. It is measured at every time-step as the absolute difference in degrees with respect to the shortest path.

#### 4.4.4. Computation time

The final measure used for evaluating the performance of each method is the computation time. The computation time is measured as the amount of time required to run a single episode (run on Apple M2 with 8 GB RAM). This is an important parameter, as it relates to the amount of training that can be done in a finite time window.

## 5. Results & discussion

### 5.1. Total rewards

Fig. 6 shows the evolution of the episodic rewards for the different methods. This figure clearly shows that the LSTM based methods using the Descending and Random sorting strategies obtain lower rewards than the other methods, highlighting the importance of the sorting strategy used for the input sequence to LSTM based methods. For comparison, when the Ascending sorting strategy is used, the total sum of rewards for the LSTM based method is comparable to the highest scoring Attention based methods, ADD and CA attention.

It is also shown that LSTM-based methods under the appropriate sorting conditions (Ascending LSTM) outperform SD attention. This is in agreement with the findings of Deniz et al. who showed that LSTM outperforms attention methods for input encoding on a similar task (Deniz et al., 2024). However, this clear performance gap is not visible for Ascending LSTM and the other two attention methods, with Ascending LSTM trailing behind the other methods for the first 3000 episodes of training before reaching comparable sum of rewards. To better show the distribution of episodic returns, the kernel density estimates (KDE) of the sum of rewards for the last 1000 episodes are shown in Fig. 7a. Because LSTM Descending and Random are performing significantly worse than the other methods, they have been left out of Fig. 7 for clarity, allowing more focus on the methods that are comparable in performance. From this figure it can be observed that the mode of the KDE for the ADD method is closer to zero than for the other methods, indicating consistently higher returns. Additionally, both the left and the right tail of the distributions are matching for the CA, ADD

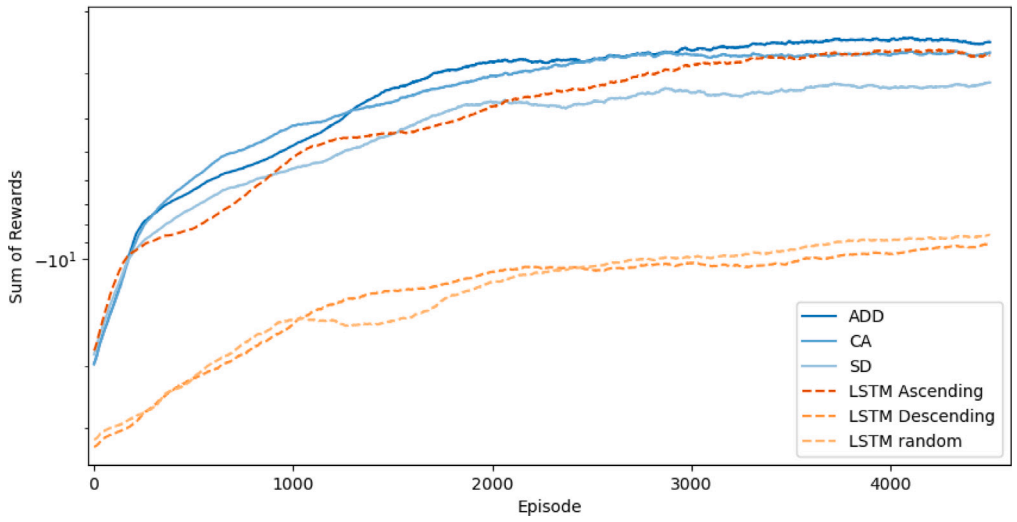



**Fig. 6.** Evolution of the rewards (moving average of window size 500) for the attention and LSTM-based methods for a fixed random seed.

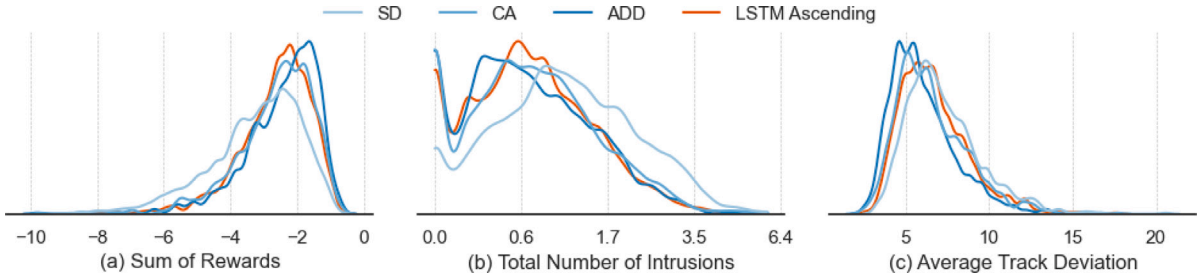



**Fig. 7.** Kernel density estimates of the (a) Sum of Rewards, (b) Total Number of Intrusions and (c) Average Track Deviation for the last 1000 episodes of the SD, CA, ADD and LSTM Ascending methods. The KDEs are generated using Scott's rule (Scott, 2015) with a bandwidth factor of 0.5. Additionally, the values are scaled using log(1+x) before generating the KDE and then converted back to the original scale to deal with the strictly positive (intrusions and track) and negative (rewards) bounds of the distributions.

and LSTM Ascending methods, with SD being a clear outlier. Because of this, the higher sum of rewards observed in Fig. 6 for ADD, can be attributed to the mode being located closer to zero, and not because of outliers or long tails of the other methods affecting their means.

To highlight these performance differences, the highest return observed over an average of 100 episode for the different methods is shown in Table 4. Here it can clearly be seen that ADD Attention obtains the highest performance, followed by CA Attention and Ascending LSTM. This challenges the hypothesis that computing the value $V$ from both the ownship's observation $X$ and the observations of other agents $Y$ leads to better performance than ADD attention.

Finally, to investigate the performance differences between the different attention methods, the experiments were rerun for three different random seeds. Fig. 8 presents the results of these experiments, and shows the median and IQR of the results. Interestingly, SD Attention has a relatively narrow IQR, which indicates that the method is relatively robust with respect to different traffic scenarios used for training. Nevertheless, the performance is almost strictly lower than that of additive attention, indicating lower asymptotic performance. The opposite is observed for CA Attention, whose spread is much wider, and is widening towards the end of training. It can also be observed that the spread is wider on both sides of the median, which suggests that the method is capable of obtaining a high performance, but is lacking stability during training. Future research should investigate if changes can be made to CA Attention to resolve this issue or whether ADD Attention should be used for Attention-based input encoding for MARL applications.

**Table 4**
Returns observed during training for the different methods (fixed random seed).

| Method | Max return (ao100) | Episode |
|---|---|---|
| Scaled dot-product attention | −2.95 | 3738 |
| Additive attention | **−2.23** | 4252 |
| Context-aware attention | −2.40 | 4503 |
| LSTM ascending | −2.41 | 4309 |
| LSTM descending | −8.56 | 4652 |
| LSTM random | −8.16 | 4498 |
| Reference values | | |
| No drift policy | 20.11 | – |
| Random policy | 21.99 | – |

### 5.2. Total number of intrusions

Fig. 9 shows the total number of intrusions observed during training for the different methods. Similar to the sum of rewards depicted in Fig. 6, the Descending and Random sorting strategies for the LSTM methods perform less effectively compared to the other methods. The remaining four methods display similar patterns, with a sharp decline in the number of intrusions during the initial training phase (first 500–1000 episodes), followed by a stabilization phase where there is minimal reduction, or even an increase, in the number of intrusions. An exception to this pattern is seen in the Ascending LSTM method, which shows two additional phases of improvement around the 2000 and 3000 episode marks, eventually converging to a similar number of intrusions as the ADD and CA attention methods. Notably, this stabilization phase reveals a significant difference between the SD attention and

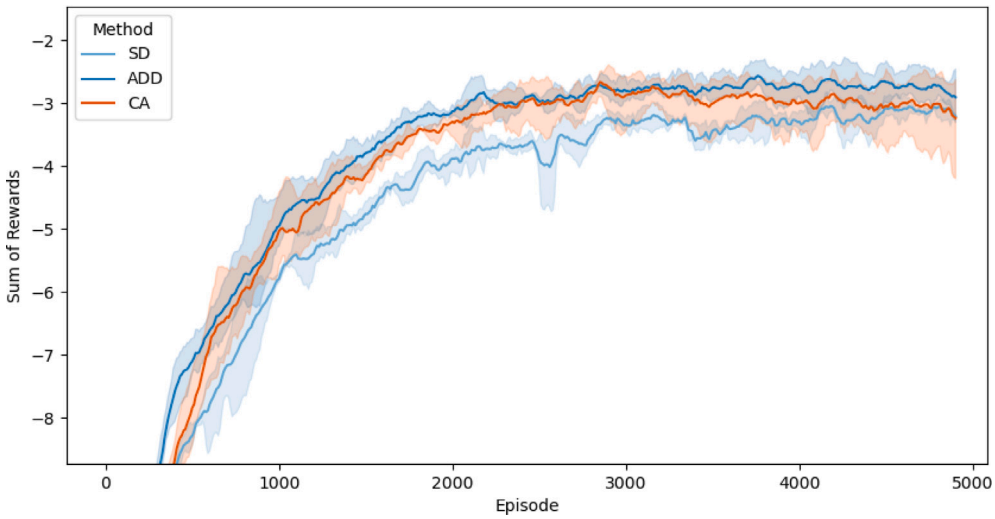



**Fig. 8.** Detail of the Median and IQR of the reward for the three attention based methods for three different seeds. Context-Aware attention in orange to highlight the larger spread associated with that method.

**Table 5**
Intrusions observed during training for the different methods (fixed random seed).

| Method | Min intrusions (ao100) | Episode |
|---|---|---|
| Scaled dot-product attention | 1.10 | 3746 |
| Additive attention | **0.61** | 2154 |
| Context-aware attention | 0.62 | 1040 |
| LSTM ascending | 0.65 | 4342 |
| LSTM descending | 2.40 | 4639 |
| LSTM random | 2.47 | 4541 |
| Reference values | | |
| No drift policy | 20.11 | – |
| Random policy | 21.99 | – |

**Table 6**
Track deviation observed during training for the different methods (fixed random seed).

| Method | Min track deviation (°) (ao100) | Episode |
|---|---|---|
| Scaled dot-product attention | 6.72 | 4719 |
| Additive attention | **5.58** | 4265 |
| Context-aware attention | 6.03 | 4734 |
| LSTM ascending | 6.29 | 4189 |
| LSTM descending | 22.57 | 4684 |
| LSTM random | 20.87 | 4020 |
| Reference values | | |
| No drift policy | 0.00 | – |
| Random policy | 54.15 | – |

the other two attention methods, as the total number of intrusions for the former is almost twice as high. Because a complete representation of the other agents' states is required to effectively reduce the number of intrusions, this suggests that SD attention is less effective at encoding the states of the other agents into the input vector when compared with the other methods. This hypothesis is further strengthened by Fig. 7b, indicated by the differences of the peaks at zero intrusions for the different methods. This shows that the number of episodes that achieve a total of zero intrusions for the SD method is less than half than that of the other listed methods.

It is possible to gather additional information by examining the lowest number of intrusions observed during training, shown in Table 5. This table shows that the lowest number of intrusions are obtained by the ADD and CA attention methods, however, these results are obtained very early on in training, at episodes 2154 and 1040 respectively. When comparing these points to the episodes of highest rewards, 4252 and 4503, it becomes clear that the methods converge to policies that prioritize lower track deviations at the cost of more intrusions. Consequently, it is difficult to determine which of the three methods – ADD Attention, CA attention, or Ascending LSTM – is best at encoding the state of the environment, as minimizing track deviation does not necessarily require effective state encoding. While this outcome is undesirable, it is an inherent consequence of modeling the reward function as a bi-criterion problem, resulting in a trade-off between safety and efficiency.

### 5.3. Average track deviation

Fig. 10 shows the average track deviation during training. In contrast with the intrusions shown in Fig. 9, the track deviation reduces more gradually, which can be explained by the initial large contribution of the number of intrusions on the total sum of rewards. Once a significant reduction in the number of intrusions is observed, the track deviation starts to decrease, as its relative contribution to the return increases. This also explains the relatively high track deviation associated with the Descending and Random LSTM methods.

Looking at the lowest obtained track deviations in Table 6, it is found that the ADD Attention method also obtain the lowest average track deviation. This is also shown in Fig. 7c, which shows a shift in the direction of zero for the KDE for the ADD method, showing that more episodes are completed with lower track deviations. This is interesting, as conflict resolution maneuvers require deviating from the path, thus it would be expected that less intrusions means lower path efficiency. However, the opposite is observed. This pattern, where path deviation is highly correlated with the number of intrusions, is observed for all of the methods, and is in line with the results of previous work (Groot et al., 2023). This seems to indicate that the effectiveness of the input encoder allows for more efficient solutions, leading to both fewer intrusions and lower track deviations. Based on this it can be concluded that ADD Attention is likely the best choice out of the evaluated methods for input state encoding for MARL applications.

### 5.4. Computation time

The final results evaluated are the computation times associated with the different methods, shown in Table 7. These results correspond with those of Vaswani et al. (2017), with SD attention performing the fastest, followed by ADD attention and LSTM as last. Considering that CA attention is identical to ADD attention with the exception of an

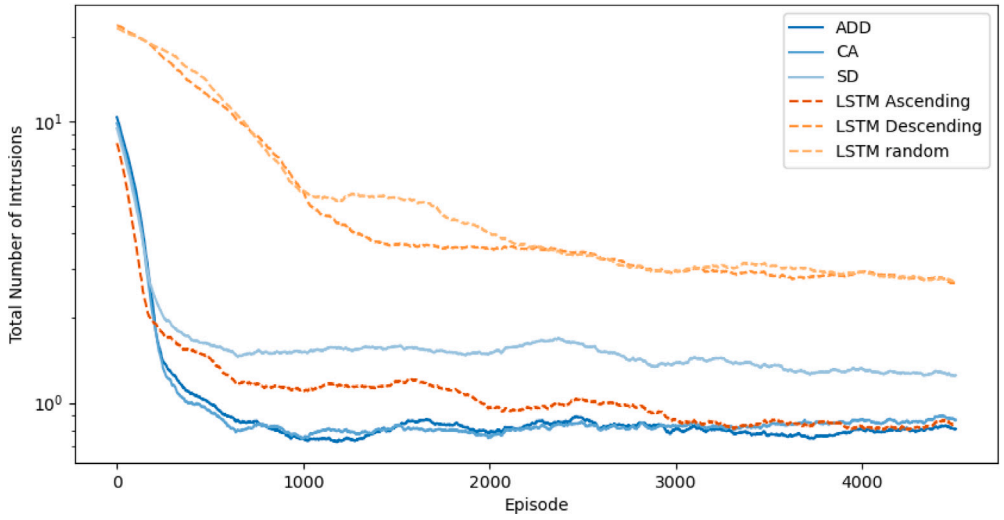



**Fig. 9.** Evolution of the total number of intrusions (averaged over 100 episodes) for the attention and LSTM-based methods for a fixed random seed.

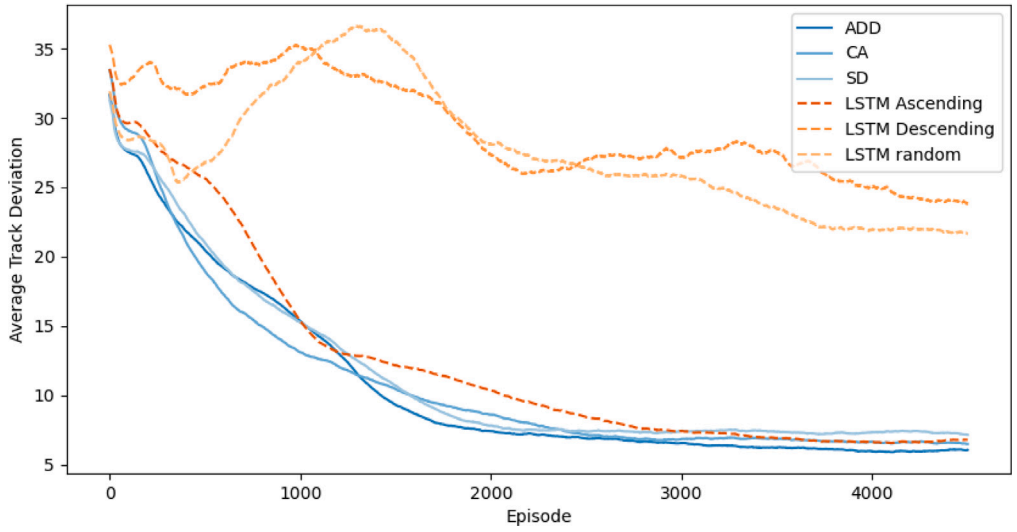



**Fig. 10.** Evolution of the average track deviation (averaged over 100 episodes) for the attention and LSTM-based methods for a fixed random seed.

additional operation to compute the Value token, it was expected that CA attention has higher computational requirements than ADD attention. Finally, there is a big difference between the attention methods and LSTM during training, which highlight an additional advantage of Attention methods over LSTM for MARL applications. Because MARL often requires many time-steps, limiting the duration required for training allows for more models to be trained in the same time frame, leading to faster development and progress in the field.

For systems with a larger number of agents, LSTM or other RNN based methods might, in theory, outperform attention based methods, considering that the computational requirements for RNN based methods scale linearly, $O(n)$, with the sequence length, whereas they scale quadratically, $O(n^2)$, for attention based methods. In practice, however, attention based methods are highly parallelizable and require minimal sequential operations, allowing them to benefit from optimized hardware more than RNN based methods (Vaswani et al., 2017). Future studies should investigate if there is indeed an inflection point in the number of agents where RNN based methods are computationally more efficient than attention based methods.

**Table 7**
Average episode duration for the different methods during training and evaluation of the models.

| Method | Training episode duration (s) | Evaluation episode duration (s) |
|---|---|---|
| Scaled dot-product attention | **11.67** | **0.46** |
| Additive attention | 13.82 | 0.50 |
| Context-aware attention | 15.18 | 0.50 |
| LSTM | 62.71 | 0.64 |
| Reference values | | |
| No drift policy | – | 0.40 |

## 6. Conclusions and future works

This paper studied the efficacy of various attention methods and LSTM strategies when applied to variable input vector length encoding in Reinforcement Learning for Conflict-Resolution in Air Traffic Control. This research mainly aimed to answer the following questions:

1. How does the performance of attention based methods compare with LSTM for the task of input encoding?
2. How much does the input sequence order for LSTM influence the final performance of the model?

To answer the first question, three different attention methods were evaluated and compared with an LSTM based input encoder: scaled dot-product (SD), additive (ADD), and a novel context-aware (CA) attention method. The performance of CA and ADD attention is comparable in all categories; total return, number of intrusions, and track deviation. However, it was found that CA attention has a larger spread in its performance when evaluated over multiple seeds, indicating that the method is less robust to variations in the training scenarios. It can therefore be worthwhile to investigate if alterations to the method can be made to improve the stability, enhancing the potential performance. Additionally, the slightly higher performance of ADD attention indicates that constructing the Value, V, using only the other agents' state is sufficient for learning a proper input encoding, while resulting in lower computational requirements.

The study also found that the maximum total return of the LSTM method (−2.41) is comparable to that of the attention methods (−2.95 SD, −2.23 ADD, −2.40 CA) when the input sequence is sorted such that the distance of the other aircraft with respect to the ownship decreases. However, for other sorting strategies, the LSTM method suffered from significant drops in maximum return (−8.16 for random sort, −8.56 for descending distance sort), showing that it is important to carefully consider the input sequence order when using LSTM as an input encoder. One limition of the study from the LSTM method in this study is the used sorting method. By sorting on distance, head on conflicts might incorrectly be classified as having a low decision importance, although the distance is rapidly decreasing. Instead of sorting on distance, future work should also study different sorting strategies, such as sorting on time until the closest point of approach, or severity of the predicted conflicts.

Based on the results, we recommend using ADD attention for input encoding in MARL-based conflict resolution. It achieves comparable safety performance to other methods – measured by the average number of intrusions (0.61 for ADD vs. 0.62 for CA and 0.65 for LSTM) – while maintaining higher path efficiency, indicated by lower average track deviation (5.58° for ADD vs. 6.03° for CA and 6.23° for LSTM). Because of this it also results in the highest overall return (−2.23 for ADD vs. −2.40 for CA and −2.41 for LSTM).

In contrast, the results for SD-attention were consistently lower, with a maximum return of −2.95, an intrusion rate of 1.10, and an average track deviation of 6.72°. One thing that was observed for all methods was that initially the total number of intrusions dropped rapidly, before going up in favor of efficiency. This could be a direct result from the used reward function, however, future research should investigate if safety guarantees can be obtained by for example using techniques used in the fields of safe RL or multi-objective RL.

As this study did not explore the influence of the number of attention heads, head size, or the number of attention blocks, it is recommended that future work investigates whether performance can be improved further and whether the ranking of methods remains consistent across different hyperparameter settings.

Furthermore, because attention-based encoders are not sequence-dependent, they offer greater robustness and lower computational costs due to parallelization (11.67 s–15.18 s per training episode, compared to 62.71 s for LSTM-based methods). This further supports the use of attention-based methods for input encoding in future applications.

Finally, because this study used a simple simulator that did not include aircraft dynamics, future studies should investigate if the methods presented in this paper also work in environments that do include dynamics and operational constraints, such as BlueSky-Gym (Groot et al., 2024b).

## CRediT authorship contribution statement

**D.J. Groot:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **J. Ellerbroek:** Writing – review & editing, Supervision, Conceptualization. **J.M. Hoekstra:** Writing – review & editing, Supervision, Investigation, Conceptualization.

## Declaration of Generative AI and AI-assisted technologies in the writing process

Statement: During the preparation of this work, the author(s) used Claude 3.5 Sonnet and ChatGPT in order to check for spelling and consistency in the usage of verbs. Additionally, these tools have been used to improve the flow of text in sections that the authors felt were difficult to read, specifically in the introduction. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

## 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.

## Appendix. Wilcoxon signed-rank test and Cohen's d effect size.

From Fig. 7a it can be observed that the distribution of the total sum of rewards is not normally distributed, but negatively skewed. Further, the samples generated are paired, e.g. the episodes/scenarios used for evaluation are the same for all methods. Because of this, the Wilcoxon signed-rank test is used for the statistical analysis for the last 1000 episodes for each of the methods.

The resulting p-values for these tests are shown in Table 8. Additionally, the Cohen's d effect size is shown in Table 9 and shows the relative differences of the magnitude. Finally, to identify consistency of the methods, Table 10 shows the percentage in how many scenarios one method outperformed the other, without looking at the magnitude of the differences.

**Table 8**
p-values resulting from the Wilcoxon signed-rank test for the last 1000 episodes for the different methods.

| | SD | ADD | CA | LSTM |
|---|---|---|---|---|
| SD | – | 6.41e−81 | 5.07e−50 | 2.54e−50 |
| ADD | 6.41e−81 | – | 7.05e−13 | 1.15e−13 |
| CA | 5.07e−50 | 7.05e−13 | – | 0.39 |
| LSTM | 2.54e−50 | 1.15e−13 | 0.39 | – |

**Table 9**
Cohen's d effect size for the last 1000 episodes for the different methods.

| | SD | ADD | CA | LSTM |
|---|---|---|---|---|
| SD | – | −0.66 | −0.47 | −0.50 |
| ADD | 0.66 | – | 0.19 | 0.19 |
| CA | 0.47 | −0.19 | – | −0.01 |
| LSTM | 0.50 | −0.19 | 0.01 | – |

**Table 10**
Percentage of episodes the total return was higher than the compared method for the last 1000 episodes for the different methods.

| | SD | ADD | CA | LSTM |
|---|---|---|---|---|
| SD | – | 31.4 | 36.6 | 36.6 |
| ADD | 68.6 | – | 55.9 | 56.6 |
| CA | 63.4 | 44.1 | – | 50.5 |
| LSTM | 63.4 | 43.4 | 49.5 | – |

## Data availability

Code is available through the github link mentioned in the paper.

## References


Achiam, Josh, Adler, Steven, Agarwal, Sandhini, Ahmad, Lama, Akkaya, Ilge, Aleman, Florencia Leoni, Almeida, Diogo, Altenschmidt, Janko, Altman, Sam, Anadkat, Shyamal, et al., 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Bahdanau, Dzmitry, Cho, Kyunghyun, Bengio, Yoshua, 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Barto, Andrew G., 2021. Reinforcement learning: An introduction. by richard's sutton. SIAM Rev. 6 (2), 423.

Brittain, Marc W., Alvarez, Luis E., Breeden, Kara, 2024. Improving autonomous separation assurance through distributed reinforcement learning with attention networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 38, pp. 22857–22863.

Brittain, Marc W., Wei, Peng, 2021. One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory. In: AIAA Scitech 2021 Forum. p. 1952.

Brittain, Marc, Yang, Xuxi, Wei, Peng, 2020. A deep multi-agent reinforcement learning approach to autonomous separation assurance. arXiv preprint arXiv:2003.08353.

Dalmau, Ramon, Allard, Eric, 2020. Air traffic control using message passing neural networks and multi-agent reinforcement learning. In: Proceedings of the 10th SESAR Innovation Days, Virtual Event. pp. 7–10.

Deniz, Sabrullah, Wu, Yufei, Shi, Yang, Wang, Zhenbo, 2024. A reinforcement learning approach to vehicle coordination for structured advanced air mobility. Green Energy Intell. Transp. 3 (2), 100157.

Groot, D.J., Ellerbroek, J., Hoekstra, J., 2023. Using relative state transformer models for multi-agent reinforcement learning in air traffic control. In: Proceedings of the 13th SESAR Innovation Days.

Groot, D.J., Ellerbroek, Joost, Hoekstra, Jacco M., 2024a. Analysis of the impact of traffic density on training of reinforcement learning based conflict resolution methods for drones. Eng. Appl. Artif. Intell. 133, 108066.

Groot, D.J., Leto, G., Vlaskin, A., Moëc, A.A.G., Ellerbroek, Joost, 2024b. BlueSky-gym: Reinforcement learning environments for air traffic applications. In: Proceedings of the 14th SESAR Innovation Days.

Haarnoja, Tuomas, Zhou, Aurick, Hartikainen, Kristian, Tucker, George, Ha, Sehoon, Tan, Jie, Kumar, Vikash, Zhu, Henry, Gupta, Abhishek, Abbeel, Pieter, et al., 2018. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

Hochreiter, S., 1997. Long short-term memory. In: Neural Computation. MIT-Press.

Hoekstra, Jacco M., van Gent, Ronald N.H.W., Ruigrok, Rob C.J., 2002. Designing for safety: the 'free flight'air traffic management concept. Reliab. Eng. Syst. Saf. 75 (2), 215–232.

Li, Sheng, Egorov, Maxim, Kochenderfer, Mykel, 2019. Optimizing collision avoidance in dense airspace using deep reinforcement learning. arXiv preprint arXiv:1912.10146.

Lowe, Ryan, Wu, Yi, Tamar, Aviv, Harb, Jean, Abbeel, Pieter, Mordatch, Igor, 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Neural Inf. Process. Syst. (NIPS).

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, et al., 2019. Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. 32.

Pham, Duc-Thinh, Tran, Ngoc Phu, Goh, Sim Kuan, Alam, Sameer, Duong, Vu, 2019. 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, pp. 1–6.

Razzaghi, Pouria, Tabrizian, Amin, Guo, Wei, Chen, Shulu, Taye, Abenezer, Thompson, Ellis, Bregeon, Alexis, Baheri, Ali, Wei, Peng, 2024. A survey on reinforcement learning in aviation applications. Eng. Appl. Artif. Intell. 136, 108911.

Ribeiro, Marta, Ellerbroek, Joost, Hoekstra, Jacco, 2022a. Distributed conflict resolution at high traffic densities with reinforcement learning. Aerospace 9 (9), 472.

Ribeiro, Marta, Ellerbroek, Joost, Hoekstra, Jacco, 2022b. Improving algorithm conflict resolution manoeuvres with reinforcement learning. Aerospace 9 (12), 847.

Scott, David W., 2015. Multivariate Density Estimation: Theory, Practice, and Visualization. John Wiley & Sons.

Undertaking, SESAR Joint, et al., 2016. European ATM Master Plan: the Roadmap for Delivering High Performing Aviation for Europe: Executive View: Edition 2015. EU: European Union.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Łukasz, Polosukhin, Illia, 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.

Wang, Zhuang, Li, Hui, Wang, Junfeng, Shen, Feng, 2019. Deep reinforcement learning based conflict detection and resolution in air traffic control. IET Intell. Transp. Syst. 13 (6), 1041–1047.

Wang, Zhuang, Pan, Weijun, Li, Hui, Wang, Xuan, Zuo, Qinghai, 2022. Review of deep reinforcement learning approaches for conflict resolution in air traffic control. Aerospace 9 (6), 294.

Wen, Han, Li, Hui, Wang, Zhuang, Hou, Xianle, He, Kangxin, 2019. Application of DDPG-based collision avoidance algorithm in air traffic control. In: 2019 12th International Symposium on Computational Intelligence and Design. ISCID, Vol. 1, IEEE, pp. 130–133.