

Transformer-based Multi-agent Reinforcement Learning for Separation Assurance in Structured and Unstructured Airspaces

Arsyi Aziz

Department of Computer Science
George Washington University
Washington D.C., United States

Peng Wei

Department of Mechanical and Aerospace Engineering
George Washington University
Washington D.C., United States

Abstract—Conventional optimization-based metering depends on strict adherence to precomputed schedules, which limits the flexibility required for the stochastic operations of Advanced Air Mobility (AAM). In contrast, multi-agent reinforcement learning (MARL) offers a decentralized, adaptive framework that can better handle uncertainty, required for safe aircraft separation assurance. Despite this advantage, current MARL approaches often overfit to specific airspace structures, limiting their adaptability to new configurations. To improve generalization, we recast the MARL problem in a relative polar state space and train a transformer encoder model across diverse traffic patterns and intersection angles. The learned model provides speed advisories to resolve conflicts while maintaining aircraft near their desired cruising speeds. In our experiments, we evaluated encoder depths of 1, 2, and 3 layers in both structured and unstructured airspaces, and found that a single encoder configuration outperformed deeper variants, yielding near-zero near mid-air collision rates and shorter loss-of-separation infringements than the deeper configurations. Additionally, we showed that the same configuration outperforms a baseline model designed purely with attention. Together, our results suggest that the newly formulated state representation, novel design of neural network architecture, and proposed training strategy provide an adaptable and scalable decentralized solution for aircraft separation assurance in both structured and unstructured airspaces.

Keywords—Advanced Air Mobility, Separation Assurance, Multi-agent Reinforcement Learning, Transformers

I. INTRODUCTION

Contemporary air traffic management (ATM) systems are built around optimization-based schedules. For instance, in the United States, Traffic Management Advisor and Time-Based Flow Management (TMA/TBFM) use schedules to sequence aircraft at metering fixes. While these scheduling approaches work well for current air traffic operations, particularly for maintaining safe aircraft separation, they may not be suitable for the emerging demands of Advanced Air Mobility (AAM). Point-to-point flights, higher traffic volumes, and the rapidly changing, stochastic airspace conditions may invalidate computed schedules, thereby reducing their capacity to address future air traffic conditions.

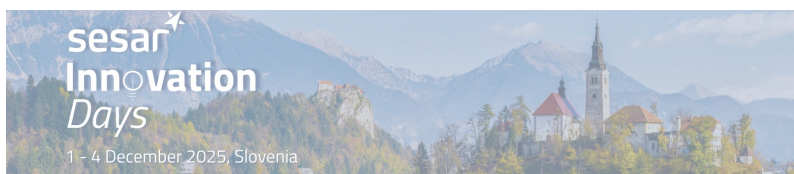
Learning-based methods, particularly reinforcement learning (RL), offer a promising alternative. Compared to schedule-based approaches, RL can learn policies that react in real time to dynamic and uncertain air traffic. In single-encounter settings, it has been shown to learn effective vectoring strategies for conflict resolution [1]. At a broader control scale, RL can be extended to multi-agent reinforcement learning (MARL), to also support cooperative, en-route separation assurance among multiple aircraft in both structured [2] and unstructured airspaces [3].

Despite recent advances, generalization capability remains a key barrier to MARL in this domain. Models trained on a narrow set of scenarios may often overfit to specific airspace-structure parameters, which consequently reduces their robustness and efficacy as the airspace structure or traffic density changes. A MARL policy trained for one airspace structure may fail in another, and one optimized for moderate traffic may become unsafe under high density or overly conservative when traffic is light. This susceptibility to scenario or distribution shift underscores the need for methods that better generalize across airspace configurations and traffic densities.

To address this limitation, we focus on the aircraft separation assurance problem and introduce a MARL framework that improves generalization across both structured and unstructured airspaces for AAM operations. Building on the encoding capacity of transformers and the adaptability of MARL, our method encodes both ownship and intruder states as tokens within a transformer encoder, which then outputs speed advisories in the form of acceleration or deceleration recommendations. Instead of training aircraft (or agents) solely on a single route structure or fixed traffic pattern, we design a diverse training procedure that exposes a learning agent to different airspace structures and traffic densities. By exposing agents to this variety, they learn representations that transfer effectively to new airspace structures unseen during training.

Additionally, to further enhance generalization, we modify the training pipeline with changes to both the reinforcement learning problem formulation and the neural network architecture. For the problem formulation, we define a state space in relative polar coordinates to emphasize ego-centric states,

A demonstration of the single-layer transformer model performance is available at <https://youtu.be/DpwM6QDnCuU>.



and we augment the reward with a speed-incentive term. For the neural network architecture, we shift from attention to tokenization by introducing a classifier token conditioned on the ownship information.

Therefore, the main contributions of this work are:

- 1) We reformulate the reinforcement learning problem using a polar state representation and a speed-incentivized reward function;
- 2) We propose an encoder–transformer architecture with a conditioned classifier token that scales to a variable number of intruder aircraft; and
- 3) We design a training regime based on procedurally generated sector structures with varying intersection angles and traffic densities.

Together, these contributions support the generalization of separation assurance models for future AAM operations.

The remainder of this paper is organized as follows. *Section II* reviews related work. *Section III* formalizes the separation assurance problem as a Markov decision process. *Section IV* introduces our modified transformer-encoder architecture. *Section V* presents the model training details. *Section VI* reports the results of the experiments and *Section VII* provides an accompanying discussion. *Section VIII* concludes with a discussion of the broader implications of our work and proposes avenues for future research.

II. RELATED WORK

A. Transformer Networks

Transformers [4] are sequential models that operate on sequences of discrete tokens. For example, in natural language processing, tokens may correspond to words or sub-words, while in aircraft separation assurance and conflict resolution, a token represents an encoding of an aircraft’s state vector. The core of the transformer architecture is the self-attention mechanism, which allows a model to compare and combine information from all token positions. This stands in contrast to recurrent architectures such as LSTMs [5] and GRUs [6], which process tokens sequentially and often struggle either to preserve information from early positions or to parallelize computation over the sequence [4]. Encoder-only architectures such as BERT [7] build on the transformer architecture to map an input sequence to contextualized, higher-level token embeddings. These token-level representations can then be aggregated in different ways for downstream classification tasks [7], [8], for example, by introducing a dedicated classification token [7] or by applying pooling operations such as mean or max pooling over tokens [9].

B. Multi-agent Reinforcement Learning for Aircraft Separation Assurance

Multi-agent reinforcement learning (MARL) is an approach to machine learning in which a model is trained to produce desired behaviors by interacting with an environment under a prescribed reward function. In the separation assurance problem, the environment is modeled as multiple aircraft

interacting within a shared airspace. Early MARL-based approaches in this setting represented the policy with a fully connected network and used input padding to handle a variable number of aircraft [10]. This design imposes a fixed upper bound on the number of aircraft that the model can process. For example, if the input layer is configured to process features for up to ten aircraft, the model cannot accommodate scenarios with more than the prescribed ten aircraft.

Subsequent work introduced attention mechanisms to better handle varying traffic levels [2]. These models learn the relative importance of each intruding aircraft, which allows them to accommodate a changing number of aircraft. This approach was later extended with transformer architectures and further improved by encoding each intruder’s state relative to the ownship [11]. Building on these developments, our work proposes modifications that further enhance adaptability to different airspace configurations and traffic densities.

III. PROBLEM FORMULATION

We study a separation assurance problem for AAM operations in a low-altitude sector, with extensions to unstructured airspace. In this problem, we consider the scenarios in which multiple aircraft attempt to maintain different desired cruising speeds. To prevent collisions, a separation assurance model must monitor the airspace for potential conflicts and issue discrete speed advisories (reduce, hold, increase) to maintain safe separation. Because desired cruising speeds are also taken into account, when safety permits, the model should ensure that each aircraft maintains proximity to these speeds. The problem is therefore posed in a speed-controlled separation assurance setting with two goals: (i) maintain safe separation of aircraft and (ii) minimize deviation from each flight’s desired cruising speed.

A. Safety-Critical Events

We define three categories of safety-critical events that may arise within an airspace:

a) Conflict: A conflict is a predicted breach of separation minima within a specified look-ahead horizon. In other words, if aircraft maintain their current trajectories and speeds, they are expected to lose safe separation within the defined time window. In our speed-only control setting, the agent must issue acceleration or deceleration commands to mitigate these predicted infringements proactively.

b) Loss of Separation: A loss of separation (LoS) occurs when the separation minima are actually violated, meaning two or more aircraft simultaneously occupy the protected zone. Upon detection of a LoS event, the aircraft must immediately coordinate speed adjustments (either an increase or decrease) to restore separation and reduce the risk of escalation toward a near mid-air collision (NMAC).

c) Near Mid-Air Collision: A near mid-air collision (NMAC) is the most severe event, defined as two or more aircraft coming within an unsafe distance of one another. The occurrence of an NMAC indicates a failure of the air traffic control system to maintain safety within the airspace. If speed

advisories cannot fully resolve the conflict within the critical thresholds, the onboard collision-avoidance system—Traffic Alert and Collision Avoidance System (TCAS) or Airborne Collision Avoidance System X (ACAS-X)—will be activated to issue coordinated vertical resolution advisories (e.g., climb or descend) to avoid collisions.

B. Mathematical Formulation

We model sector-level coordination of multiple aircraft (agents) as a fully observable, multi-agent Markov decision process with a time-varying population. The process is described by the tuple

$$\mathcal{M} := \langle \mathcal{N}_{\max}, \mathcal{S}, \{\mathcal{A}_i\}_{i \in \mathcal{N}_{\max}}, \mathcal{P}, \mathcal{R}, \gamma \rangle,$$

where $\mathcal{N}_{\max}$ indexes a superset of possible aircraft, $\mathcal{S}$ is the state space, $\mathcal{A}_i$ is the action space of agent i , $\mathcal{P}$ is the transition kernel (determined by aircraft dynamics and atmospheric conditions), $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. Furthermore, let $\mathcal{N}_t \subseteq \mathcal{N}_{\max}$ denote the set of active aircraft at time t .

At each discrete time t , the environment has a global state $s_t \in \mathcal{S}$. From this global state, each active aircraft $i \in \mathcal{N}_t$ observes an ego-relative state

$$s_t^{(i)} := \psi_i(s_t) \in \mathcal{S}_i,$$

where $\psi_i : \mathcal{S} \rightarrow \mathcal{S}_i \subset \mathcal{S}$ is a deterministic, agent-specific projection that extracts the states of aircraft i from the global state expressed in i 's relative frame. ADS-B and error-free communications assume these states are accurate.

Each aircraft then selects an action according to a policy

$$a_t^{(i)} \sim \pi_i(\cdot \mid s_t^{(i)}),$$

and we write the joint action profile as the set of all actions $a_t := \{a_t^{(i)}\}_{i \in \mathcal{N}_t}$. The next state evolves as

$$s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t),$$

which implicitly allows for arrivals/departures of aircraft, so that the number of aircraft may change and $\mathcal{N}_{t+1}$ need not equal $\mathcal{N}_t$ in general.

Following each transition, aircraft i receives a reward

$$r_t^{(i)} := \mathcal{R}_i(s_t^{(i)}, a_t^{(i)}).$$

We denote $r_t := \{r_t^{(i)}\}_{i \in \mathcal{N}_t}$ as the collection of all rewards obtained at timestep t . A trajectory is denoted as $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T)$, defined over a finite horizon T . Completing a trajectory, for agent i , the expected discounted return is

$$J_i(\pi) := \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t r_t^{(i)} \right].$$

Since we follow a general-sum multi-agent setting, we maximize the average of these values across all aircraft, so the overall objective is to find the maximizing policy

$$\pi^* = \arg \max_{\pi} \frac{1}{|\mathcal{N}_{\max}|} \sum_{i \in \mathcal{N}_{\max}} J_i(\pi).$$

C. State Space

We construct the agent's state space in an egocentric frame, consisting of both ownship features and a set of intruder features.

1) Ownship Features

The ownship features provide the agent with speed-related information, consisting of the following elements:

- *Calibrated airspeed*: the airspeed of aircraft i at time t , corrected for instrument and positional errors, denoted by $v_{\text{cas},t}^{(i)}$.
- *Speed deviation*: the absolute difference between the calibrated airspeed and the desired speed of aircraft i at time t , defined as

$$\Delta v_t^{(i)} := \left| v_{\text{cas},t}^{(i)} - v_{\text{des}}^{(i)} \right|.$$

2) Intruder Features

The intruder features are the set of intruder information encoded relative to the ownship, which includes:

- *Relative position*: the position of each intruder relative to the ownship, expressed in polar coordinates. Let $d_t^{(i,j)}$ denote the Euclidean distance between aircraft i and j at time t . The relative position encompasses four components:

- *Distance to the near mid-air collision (NMAC) boundary*:

$$d_{\text{nmac},t}^{(i,j)} := d_t^{(i,j)} - r_{\text{nmac}},$$

where r_{nmac} denotes the NMAC radius of 500 feet.

- *Distance to loss of separation (LOS) boundary*:

$$d_{\text{pz},t}^{(i,j)} := d_t^{(i,j)} - r_{\text{pz}}^{(i)},$$

where $r_{\text{pz}}^{(i)}$ is the protection zone radius for loss of separation of aircraft i , which is by default set to 5 nautical miles.

- *Relative bearing angle of intruder j with respect to ownship i* :

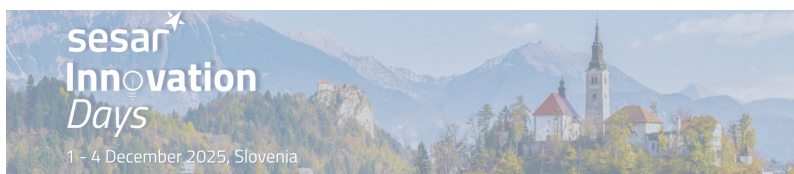
$$\theta_t^{(i,j)} := (\arctan2(\Delta y_t^{(i,j)}, \Delta x_t^{(i,j)}) - \phi_t^{(i)}).$$

Here, $\Delta x_t^{(i,j)}$ and $\Delta y_t^{(i,j)}$ are the relative position components of intruder j with respect to ownship i . The function $\arctan2$ is the four-quadrant inverse tangent, which returns the absolute bearing of the intruder to the ownship. Subtracting the ownship heading $\phi_t^{(i)}$ yields the relative bearing. This feature is encoded using the sine and cosine functions to eliminate angular discontinuities in the features.

- *Loss-of-separation indicator*:

$$b_{\text{los},t}^{(i)} := \mathbb{1}_{\{d_t^{(i,j)} \leq r_{\text{pz}}\}},$$

a binary variable that equals 1 if aircraft i and j are within the protection zone and 0 otherwise. Although this condition can be inferred from $d_t^{(i,j)}$



relative to $r_{pz}^{(i)}$, the explicit inclusion of the indicator serves two purposes: (i) it provides a direct encoding of loss-of-separation events, and (ii) it allows the model to selectively deactivate or modulate other features when such loss-of-separation events occur.

- *Relative velocity.* In addition to positional features, we also include two velocity features, namely radial velocity and tangential velocity. To define these features, at time t , denote the relative position and velocity between ownship i and intruder j as

$$\mathbf{p}_t^{(i,j)} := \begin{bmatrix} \Delta x_t^{(i,j)} \\ \Delta y_t^{(i,j)} \end{bmatrix}, \quad \mathbf{v}_t^{(i,j)} := \begin{bmatrix} \Delta v_{x,t}^{(i,j)} \\ \Delta v_{y,t}^{(i,j)} \end{bmatrix},$$

where $\Delta x_t^{(i,j)} = x_t^{(j)} - x_t^{(i)}$ and $\Delta y_t^{(i,j)} = y_t^{(j)} - y_t^{(i)}$ are the east-west and north-south position differences, and $\Delta v_{x,t}^{(i,j)} = v_{x,t}^{(j)} - v_{x,t}^{(i)}$, $\Delta v_{y,t}^{(i,j)} = v_{y,t}^{(j)} - v_{y,t}^{(i)}$ are the corresponding velocity-component differences.

The radial unit vector, pointing from i to j , is

$$\hat{\mathbf{e}}_{p,t}^{(i,j)} := \frac{\mathbf{p}_t^{(i,j)}}{\|\mathbf{p}_t^{(i,j)}\|},$$

and the tangential unit vector is obtained by a counterclockwise rotation of $\pi/2$:

$$\hat{\mathbf{e}}_{\psi,t}^{(i,j)} := R_{\pi/2} \hat{\mathbf{e}}_{p,t}^{(i,j)}.$$

Projecting the relative velocity onto these axes yields the two features:

– *Radial velocity:*

$$v_{p,t}^{(i,j)} := \mathbf{v}_t^{(i,j)} \cdot \hat{\mathbf{e}}_{p,t}^{(i,j)},$$

– *Tangential velocity:*

$$v_{\psi,t}^{(i,j)} := \mathbf{v}_t^{(i,j)} \cdot \hat{\mathbf{e}}_{\psi,t}^{(i,j)}.$$

D. Action Space

At each decision step the agent selects one of three speed advisories: increase, decrease, or hold the current calibrated airspeed. Adjustments are applied in fixed increments of 5 knots. Since we consider the aircraft to be AAM vehicles with hovering capabilities, the calibrated airspeed is bounded between 0 knots and the aircraft's maximum permitted speed. We denote endpoints of this range as $v_{\min} = 0$ and $v_{\max}$.

E. Reward Function

To train a policy that issues speed advisories toward aircraft desired speeds while avoiding safety-critical events, we design a reward function that balances positive and negative components. The positive terms reinforce progress toward desirable behavior, whereas the negative terms penalize unsafe or undesirable outcomes. Importantly, the reward is carefully designed to promote safe and adaptive behavior that remains effective in environments beyond those seen during training.

Let $\alpha_{(\cdot)}$ denote a hyperparameter of a reward component. The reward components are then categorized into three groups:

1) No Conflict:

- a) *Proximity to desired speed:* If no conflict is expected to occur within a fixed look-ahead horizon L , the agent is rewarded for approaching the desired speed:

$$\mathcal{R}_v(s_t, a_t) := (+) \alpha_v \Delta \hat{v}_t,$$

where $\Delta \hat{v}_t = 1 - \frac{\Delta v_t}{v_{\max} - v_{\min}} \in [0, 1]$ is the normalized proximity to desired speed.

2) In Conflict:

- a) *Time to intrusion:* If a conflict is expected to occur within a fixed look-ahead horizon L , the agent incurs a penalty proportional to the minimum normalized time-to-intrusion:

$$\mathcal{R}_{\text{conflict}}(s_t, a_t) := (-) \alpha_{\text{conflict}} \hat{T}_{\text{los},t},$$

where

$$\hat{T}_{\text{los},t} := \text{clip} \left(\left[\frac{L - \min_{j \in \mathcal{N} \setminus \{i\}} T_{\text{los},t}^{(i,j)}}{L} \right]; 0, 1 \right),$$

and $T_{\text{los},t}^{(i,j)} \in [0, L]$ denotes the extrapolated time to loss of separation between aircraft i and j , computed from their relative velocity components.

- b) *Distance to NMAC:* If the agent is already in loss of separation, we impose an additional penalty based on the normalized distance to the nearest intruder:

$$\mathcal{R}_{\text{los}}(s_t, a_t) := (-) \alpha_{\text{los}} \hat{d}_t^{(i,j)},$$

where the normalized distance between aircraft i and j is defined as

$$\hat{d}_t^{(i,j)} := \text{clip} \left(\left[\frac{r_{\text{pz}} - \min_{j \in \mathcal{N} \setminus \{i\}} (d_t^{(i,j)})}{r_{\text{pz}} - r_{\text{nmac}}} \right]; 0, 1 \right),$$

and $d_t^{(i,j)}$ denotes the Euclidean distance between aircraft i and j .

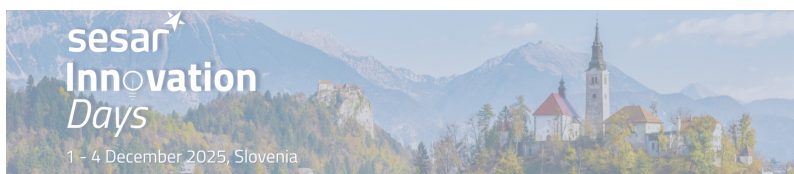
- 3) *In NMAC:* If an agent is in a near-mid-air collision, we apply a large penalty to disincentivize this behavior:

$$\mathcal{R}_{\text{nmac}} := (-) \alpha_{\text{nmac}}.$$

In our setting, we set α_{nmac} to -100 .

By aggregating the individual reward components, the overall reward function takes the compact form:

$$\mathcal{R}(s_t, a_t) = \begin{cases} \mathcal{R}_v(s_t, a_t), & \text{if **not in** conflict,} \\ \mathcal{R}_{\text{conflict}}(s_t, a_t) + \mathcal{R}_{\text{los}}(s_t, a_t), & \text{if **in** conflict,} \\ \mathcal{R}_{\text{nmac}}, & \text{if **in** NMAC.} \end{cases}$$



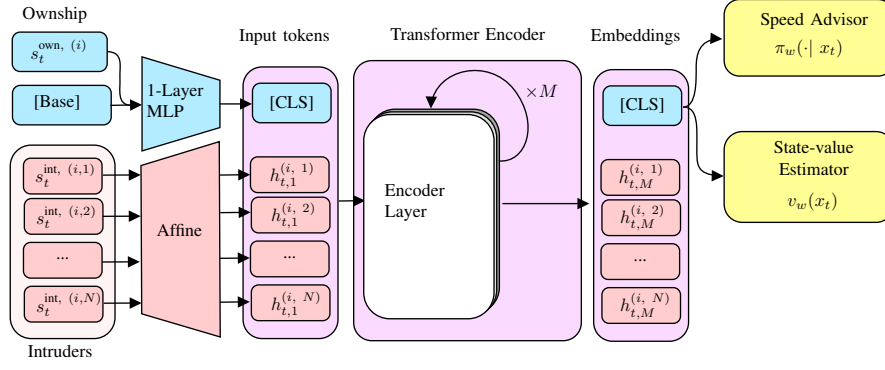


Figure 1. The implemented network adopts an encoder–transformer architecture, where intruder information is represented as intruder tokens. A classifier ([CLS]) token, derived from ownship information, is appended to this set. The complete token sequence is processed by a transformer encoder with M layers. The final-layer [CLS] token is used to produce both the policy and the state-value estimates.

IV. NEURAL NETWORK ARCHITECTURE

Figure 1 depicts our neural network architecture. It builds on an encoder-transformer design from natural language processing [7], adapted to operate on intruder tokens rather than word tokens. In this design, ownship information is encoded by a conditioned classifier token that aggregates information from the intruder tokens. We detail each component as follows:

Ownship feature adaptation: The ownship features are encoded as a conditioned classifier token. To construct this token, we first define a base token as a learnable parameter initialized from a standard Gaussian distribution. This learnable prior is then fused with the ownship information through concatenation, followed by an affine transformation, GELU activation [12], and layer normalization [13] to explicitly condition the token on the ego state. The resulting representation serves as the classifier token, denoted as [CLS], which will later serve as an aggregator of information across the token space.

Intruder feature adaptation: The intruder features are encoded as intruder tokens. To obtain these tokens, we individually transform the features of each intruder into the embedding space through an affine transformation with layer normalization. This process produces a set of intruder tokens, one for each intruder.

Token processing and aggregation: After adapting both the ownship and intruder features to the embedding space, we concatenate the [CLS] token with the intruder tokens to form a token set. This set is then processed by an M -layer transformer encoder, which applies self-attention [4], layer normalization, and GELU activations to produce an attention-refined token set. From the encoder output, we extract the [CLS] token and use it as input to both the policy distribution (speed advisor) and the state-value function estimator heads.

Here we explicitly use the [CLS] token as a global aggregator rather than pooling over intruder tokens, for three reasons. First, it allows the encoder to gather all intruder information into a single representation. This removes the need for ad hoc pooling schemes over intruder tokens, such as min- or max-pooling. Second, because the [CLS] token is conditioned on ownship features, it carries that context through all encoder layers. Alternative designs would need to

inject ownship information later at the transformer heads via concatenation, which may restrict ownship context to the final stages of processing and weaken its influence on earlier, lower-level representations. Third, this design naturally supports a variable number of intruders, including the zero-intruder case, without requiring explicit padding across intruder tokens.

V. MODEL TRAINING

This section outlines our training methodology. We begin with a brief overview of the proximal policy optimization algorithm used to train our models, followed by a description of the training environment and then the training details.

A. Proximal Policy Optimization

We train our network using the clipped variant of Proximal Policy Optimization (PPO) [14]. PPO extends the policy gradient framework, where the parameters of a stochastic policy $\pi_w(a | s)$ are optimized by ascending the gradient of the expected return. In standard policy gradient methods, the update direction is given by

$$\nabla_w J(w) = \mathbb{E}_t \left[\nabla_w \log \pi_w(a_t | s_t) G_t \right],$$

where G_t is the return-to-go and $\mathbb{E}_t$ denotes an expectation over time steps collected under the current policy. This update increases the likelihood of actions with higher returns and decreases it for less beneficial actions.

Large steps in this naive update, however, can destabilize the learning process. PPO addresses this issue by maximizing a surrogate objective that constrains the deviation between the new and old policies. This surrogate objective is defined as:

$$\max_w \hat{\mathbb{E}} \left[\min(r_t(w) \hat{A}_t, \text{clip}(r_t(w), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right],$$

where $r_t(w) = \frac{\pi_w(a_t | s_t)}{\pi_{w_{\text{old}}}(a_t | s_t)}$ is the probability ratio between the updated and old policies, $\hat{A}_t$ is an estimate of the advantage function (typically computed as the difference between the obtained reward R_t and a learned state-value function $v(s_t)$), and ϵ is the clipping parameter. The clipping parameter ϵ prevents overly aggressive updates and leads to more stable training.

B. Training Environment

We train our policy in a simulated air traffic control environment built upon the BlueSky simulator [15]. The environment models a circular flight sector in which two routes intersect at a central point. Each route extends over a fixed distance with a sector radius of approximately 30 nautical miles. Agents act at one-second intervals. To improve generalization, we randomize the intersection angle between the routes and enforce a minimum separation of 5 nautical miles between the route endpoints. A visualization of the training environment is provided in Figure 2.

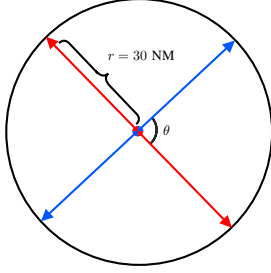


Figure 2. The simulated training environment consists of two routes whose endpoints are uniformly sampled within a circle of radius 30 nautical miles. The routes may intersect at angles $\theta \in [0^\circ, 180^\circ]$ and can be traversed in either direction.

Unlike traditional fixed-wing settings, we consider aircraft with VTOL capabilities, which have both forward motion and the ability to hover. This additional degree of freedom not only modifies the state dynamics but also introduces the challenge of hovering behavior, which may lead to deadlock (i.e., the environment may not terminate).

To ensure that the environment terminates, we consider two conditions under which aircraft may be removed from the environment. The first occurs when an aircraft takes too long to reach its destination. For this, we precompute an expected time of arrival for each aircraft based on its desired speed, plus a 20-minute buffer, which we set due to battery constraints. If an aircraft exceeds this precomputed time, it is removed from the environment. The second scenario arises in the event of a near mid-air collision. In such a case, the infringing aircraft are removed and a substantial penalty (see Section III-E) is imposed.

C. Training Details

We train the network described in Section IV using PPO across eight parallel environments, collectively simulating approximately 75 days of traffic. A fixed-length rollout buffer aggregates trajectories from all agents, with each environment generating sequences of 4096 seconds. Transitions are randomly shuffled and sampled to optimize the PPO objective. To stabilize training, we apply value clipping and advantage normalization.

In total, 200 updates are performed on a transformer-encoder network with an embedding dimension of 128 and an encoder feed-forward dimension of 512. We further train three model variants with encoder depths of 1, 2, and 3 layers. To

TABLE I. THE TRANSFORMER-BASED NEURAL NETWORK CONFIGURATIONS EMPLOYED IN OUR EXPERIMENTS.

Component	Setting
Encoder layers	[1, 2, 3]
Token/embedding dimension	128
Feed-forward (FFN) dimension	512
Attention heads	16

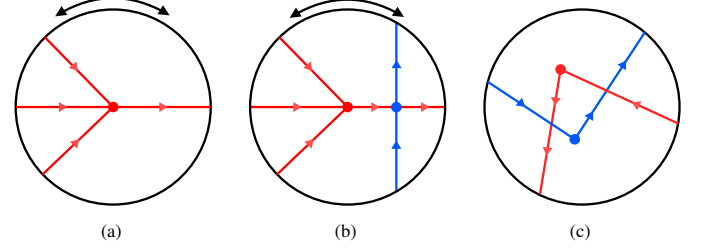


Figure 3. Evaluation of the model under three scenarios simulating structured and unstructured airspace. Case (a) and Case (b) represent structured airspace, but with rotational variability. Case (c) serves as a proxy for unstructured airspace, which includes one intermediary waypoint. For visualization purposes, we only show two routes in Case (c). However, the possible number of routes is dictated by the spawning scheme defined in Table III.

ensure fair evaluation and reduce sensitivity to initialization, each model is trained three times with different random seeds. A summary of the relevant neural network configurations and hyperparameters is provided in Table I and Table II, respectively.

TABLE II. HYPERPARAMETERS EMPLOYED IN THE PPO TRAINING OF THE TRANSFORMER NETWORK.

Hyperparameter	Value
Number of updates	200
Parallel environments	8
Time horizon (T)	4096
Batch size	128
Update epochs	4
Discount factor (γ)	0.99
GAE- λ	0.95
Clip coefficient (ϵ)	0.2
Entropy coefficient	0.01
Value function coefficient	0.5
Maximum gradient norm	0.5
Advantage normalization	✓
Value function clipping	✓

VI. SIMULATION EXPERIMENTS

This section presents the results of our simulation experiments. In this section, we will first examine the training curves to compare the three neural network configurations, each with 1, 2, and 3 encoder layers. After that, we will assess each model's ability to adapt to unseen scenarios in both structured and unstructured airspaces. Lastly, we will compare our transformer-based architecture with a neural network architecture constructed solely with attention mechanisms.

A. Training Curves

We trained three transformer configurations with 1, 2, and 3 encoder layers for 200 updates, running three independent runs

per configuration. Throughout training, we evaluated model performance using two metrics:

- 1) λ -return: the bootstrapped return as calculated by the Generalized Advantage Estimation (GAE) [16]; and
- 2) Policy entropy: a measure of the stochasticity (i.e., predictability) of the learned policies.

We begin with Figure 4a, which plots the bootstrapped returns over the 200 training updates. In this plot, we observe that all encoder configurations show a steady rise in λ -returns over time. The curves start near 0 and rise to about 75 to 83 across configurations. Among them, the 1-layer encoder attains the highest final λ -return at 83.1, followed by the 2-layer at 78.8, and the 3-layer at 75.1.

The entropy of the policies is shown in Figure 4b. Across all transformer network configurations, entropy decreases over time, reflecting convergence to lower-entropy policies. The 1-layer encoder exhibits a marked deviation between roughly the 25th and 160th updates, where entropy rises and stabilizes near 0.75 before returning to a lower value. The 2-layer encoder shows the broadest range across runs, with a high upper bound of about 0.8 that emerges toward the end of training. In contrast, the 3-layer encoder displays a steadier decline with tighter variability. By the final updates, all models converge to comparable entropy values, ranging from around 0.35 to 0.5.

B. Unseen Sector Structure Configurations

After training the three different transformer configurations on the training environment (see Figure 2), we then evaluated the trained model's performance on three unseen airspace structures, as visualized in Figure 3:

- Case (a): a single merge point where three routes converge;
- Case (b): a single merge point of three routes with a downstream four-way intersection; and
- Case (c): an airspace with procedurally generated routes whose endpoints are sampled on the sector boundary with one intermediate waypoint placed randomly within the sector.

The first two cases represent structured airspace, whereas the third serves as a proxy for unstructured airspace. To assess the generalizability of the model and to ensure rotational invariance, Case (a) and Case (b) are randomly rotated by an angle uniformly sampled from 0° to 360°. Because Case (c) includes randomly sampled intermediate waypoints, some instances may be infeasible without requiring all aircraft to hover. To resolve this issue, we allow the early termination of an aircraft's trajectory if it is in a loss-of-separation and can decelerate to a stop. Additional details regarding the sampling distributions of the aircraft are provided in Table III.

C. Performance on Unseen Sector Structures

To analyze model performance in unseen sector configurations, we focus on three metrics: (i) the number of near mid-air collisions (NMACs), (ii) the cumulative time in loss of separation (LOS) across all aircraft pairs, summed over

TABLE III. AIRCRAFT DISTRIBUTIONS IN EACH OF THE ENVIRONMENTS.

Environment	N aircraft	Spacing (s)	Desired speed (kts)
Training	$U(1, 20)$	$U(60, 1200)$	$U(60, 120)$
Case (a)	$U(1, 20)$	$U(60, 1200)$	110
Case (b)	$U(1, 20)$	$U(60, 1200)$	110
Case (c)	$U(1, 10)$	$U(60, 1200)$	$U(60, 120)$

† Here, N aircraft denotes the number of aircraft spawned in the flight sector. $U(a, b)$ denotes a draw from the uniform distribution on $[a, b]$.

all timesteps, and (iii) adherence to desired speeds, measured as the proportion of timesteps in which aircraft are within 10 knots of their desired cruising speed. Benchmark results are summarized in Table IV, averaged across 1,000 evaluation episodes with three independently trained sets of weights for each neural network configuration.

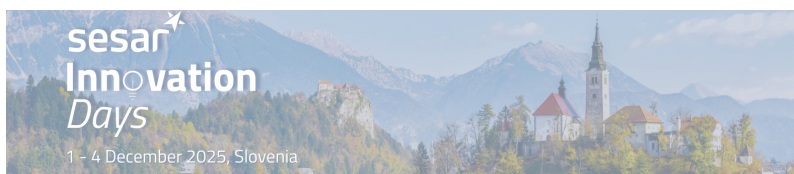
We begin by analyzing the model's capability to prevent NMACs. As shown in Table IV, the different transformer configurations indicate relatively safe separation capabilities. In particular, our 1 and 2 encoder layer networks show near-zero NMACs, each with average incursion values of 0.002 and 0.001, respectively. By contrast, the three-layer network incurs a slightly higher occurrence, with an average number of NMACs of 0.037.

Continuing to LoS, we observe that all transformer configurations generally incur fewer LoS events in the structured airspace of Case (a) and Case (b), while showing noticeably higher incursions in the unstructured airspace of Case (c). For the 1-layer encoder, the overall mean LoS value is 678.154, with substantially lower averages of 30.327 in Case (a) and 96.785 in Case (b). The 2-layer encoder reaches an overall mean LoS of 1152.971, driven upward by the high average of 2002.420 in Case (c). The 3-layer encoder averages 966.157 overall and performs worst in the structured settings of Case (a) and Case (b). We note that in all scenarios the aircraft are restricted to speed adjustments only, which likely limits their ability to mitigate safety incursions, particularly in Case (c).

Lastly, in terms of adherence to the desired cruising speeds, the three-layer encoder attains the closest desired speed adherence with a proportion of 72%. Meanwhile, the one and two-layer encoders have a slightly lower value of 69.8% and 65.2%, respectively. This reduced proportion may be attributed to the higher safety factors (lower NMACs and LoS), which prompt the aircraft to decelerate or accelerate more often to prevent incursions from occurring.

D. Comparisons to Pure Attention

We compare our approach with a pure attention model [2]. As summarized in Table IV, our transformer-based neural network, particularly the single-layer encoder configuration, achieves a higher level of safety with fewer NMACs and generally shorter time in LoS. In addition, all our transformer network configurations with 1, 2, and 3 encoder layers tracked the desired cruising speed more closely than the pure attention baseline.



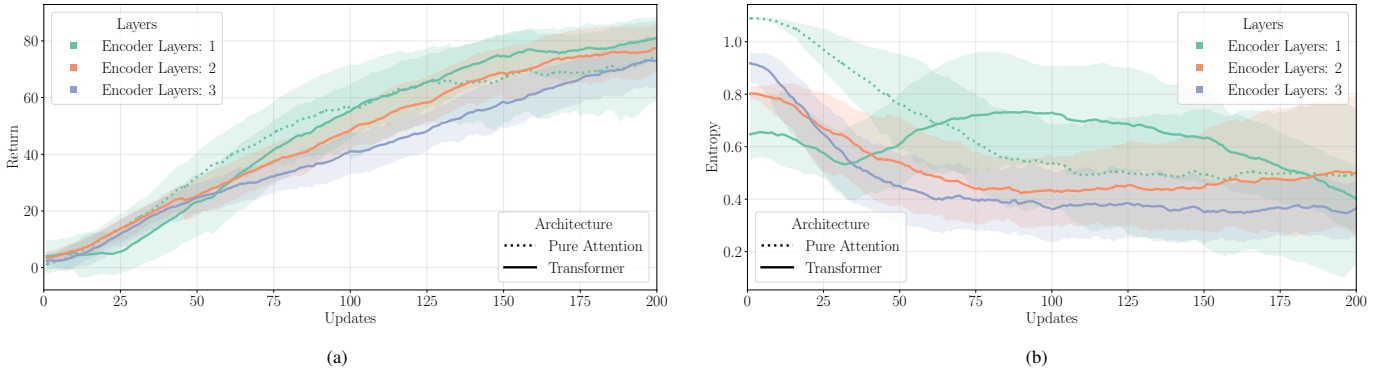


Figure 4. The average λ -returns (a) and entropy (b) of the transformer network during training, shown for configurations with 1, 2, and 3 encoder layers. Each configuration was trained using three random seeds. Solid lines represent the mean values, while shaded regions indicate the range between the minimum and maximum values. Lines are smoothed with an exponential moving average with $\alpha = 0.05$.

TABLE IV. PERFORMANCE AND SAFETY METRICS OF THE DIFFERENT NETWORK CONFIGURATIONS ACROSS 1000 EPISODES.

Environment	Density	Pure Attention			1-Layer Transformer			2-Layer Transformer			3-Layer Transformer		
		NMACs	Speed Adher.	LoS Time	NMACs	Speed Adher.	LoS Time	NMACs	Speed Adher.	LoS Time	NMACs	Speed Adher.	LoS Time
Training	≤ 5	0.008 ± 0.125	0.653 ± 0.175	108.338 ± 527.907	0.000 ± 0.000	0.843 ± 0.182	119.300 ± 570.666	0.000 ± 0.000	0.819 ± 0.170	190.182 ± 813.784	0.011 ± 0.145	0.868 ± 0.132	92.371 ± 461.763
	6-10	0.080 ± 0.437	0.578 ± 0.205	1771.115 ± 4060.318	0.008 ± 0.124	0.709 ± 0.212	1119.789 ± 2463.339	0.002 ± 0.067	0.646 ± 0.235	2215.736 ± 4111.380	0.049 ± 0.329	0.726 ± 0.202	1017.803 ± 2379.765
	11+	0.337 ± 0.927	0.371 ± 0.218	11727.380 ± 14699.918	0.000 ± 0.000	0.490 ± 0.206	4059.592 ± 5150.277	0.000 ± 0.000	0.418 ± 0.217	6455.796 ± 6595.136	0.091 ± 0.417	0.539 ± 0.218	3161.449 ± 4144.970
	Average	0.081 ± 0.453	0.583 ± 0.211	2191.441 ± 6422.559	0.005 ± 0.097	0.736 ± 0.223	1016.157 ± 2585.519	0.001 ± 0.052	0.685 ± 0.241	1874.085 ± 4020.136	0.040 ± 0.294	0.756 ± 0.206	898.693 ± 2344.195
Case (a)	≤ 5	0.000 ± 0.000	0.671 ± 0.110	3.524 ± 43.188	0.000 ± 0.000	0.828 ± 0.081	0.515 ± 5.325	0.000 ± 0.000	0.789 ± 0.128	3.973 ± 19.792	0.000 ± 0.000	0.858 ± 0.082	131.431 ± 804.446
	6-10	0.001 ± 0.045	0.645 ± 0.138	55.615 ± 326.128	0.000 ± 0.000	0.759 ± 0.075	13.360 ± 145.403	0.001 ± 0.045	0.696 ± 0.152	97.871 ± 650.170	0.010 ± 0.143	0.781 ± 0.071	387.216 ± 1531.111
	11+	0.014 ± 0.166	0.560 ± 0.169	650.686 ± 1550.735	0.002 ± 0.070	0.572 ± 0.111	78.825 ± 192.643	0.000 ± 0.000	0.542 ± 0.175	744.886 ± 2412.845	0.063 ± 0.363	0.660 ± 0.075	1544.828 ± 3230.180
	Average	0.005 ± 0.097	0.622 ± 0.152	225.269 ± 916.817	0.001 ± 0.037	0.712 ± 0.123	30.327 ± 158.063	0.001 ± 0.037	0.659 ± 0.175	273.182 ± 1412.005	0.025 ± 0.230	0.750 ± 0.095	713.279 ± 2217.769
Case (b)	≤ 5	0.000 ± 0.000	0.677 ± 0.114	0.000 ± 0.000	0.000 ± 0.000	0.839 ± 0.066	0.000 ± 0.000	0.000 ± 0.000	0.814 ± 0.094	3.889 ± 18.900	0.000 ± 0.000	0.868 ± 0.047	4.791 ± 25.383
	6-10	0.002 ± 0.066	0.593 ± 0.148	156.961 ± 509.802	0.000 ± 0.000	0.700 ± 0.080	37.113 ± 113.309	0.000 ± 0.000	0.641 ± 0.129	206.468 ± 863.310	0.019 ± 0.194	0.731 ± 0.082	249.403 ± 895.354
	11+	0.023 ± 0.216	0.496 ± 0.159	1245.955 ± 1747.369	0.000 ± 0.000	0.551 ± 0.079	215.158 ± 408.550	0.000 ± 0.000	0.482 ± 0.125	962.233 ± 2023.816	0.177 ± 0.594	0.599 ± 0.081	1271.186 ± 2338.114
	Average	0.010 ± 0.141	0.558 ± 0.159	555.713 ± 1250.620	0.000 ± 0.000	0.652 ± 0.110	96.785 ± 268.506	0.000 ± 0.000	0.590 ± 0.151	462.197 ± 1418.890	0.073 ± 0.390	0.688 ± 0.105	598.839 ± 1626.276
Case (c)	≤ 3	0.000 ± 0.000	0.535 ± 0.210	149.869 ± 527.145	0.000 ± 0.000	0.771 ± 0.179	223.657 ± 679.895	0.000 ± 0.000	0.759 ± 0.150	306.918 ± 802.779	0.000 ± 0.000	0.774 ± 0.153	217.779 ± 659.005
	4-6	0.014 ± 0.166	0.537 ± 0.179	1415.154 ± 1793.011	0.003 ± 0.078	0.688 ± 0.109	1480.423 ± 1883.388	0.000 ± 0.000	0.667 ± 0.104	1996.577 ± 2150.875	0.009 ± 0.135	0.678 ± 0.108	1602.484 ± 2009.602
	7+	0.018 ± 0.191	0.522 ± 0.179	3525.313 ± 3123.919	0.009 ± 0.132	0.614 ± 0.093	3626.186 ± 2983.981	0.000 ± 0.000	0.600 ± 0.085	4420.622 ± 3195.953	0.018 ± 0.186	0.617 ± 0.083	3681.700 ± 3049.316
	Average	0.012 ± 0.154	0.534 ± 0.185	1485.924 ± 2133.126	0.003 ± 0.082	0.692 ± 0.132	1569.346 ± 2183.789	0.000 ± 0.000	0.676 ± 0.122	2002.420 ± 2447.275	0.009 ± 0.131	0.687 ± 0.125	1653.816 ± 2273.653
Overall	Average	0.027 ± 0.256	0.574 ± 0.181	1114.587 ± 3556.437	0.002 ± 0.066	0.698 ± 0.156	678.154 ± 1817.575	0.001 ± 0.032	0.652 ± 0.181	1152.971 ± 2676.009	0.037 ± 0.279	0.720 ± 0.143	966.157 ± 2173.696

† Here, density refers to the maximum number of aircraft appearing in the sector at any point in an episode, while speed adherence refers to the proportion of time in which aircraft are within 10 knots of their desired speed. Cells highlighted in green indicate the best overall performance, while cells highlighted in orange indicate the worst overall performance. The configuration in bold indicates the best model.

VII. DISCUSSIONS

We study safe-separation assurance with two objectives: (i) maintaining safe separation and (ii) minimizing deviation from the desired cruising speed. We cast the problem in a multi-agent reinforcement learning setting and introduce a reformulated state space and neural network architecture. To improve generalization, we trained the models on varied airspace structures with different intersection angles and traffic densities. Our neural network architecture is a transformer network containing a [CLS] token conditioned on ownship features. Furthermore, we benchmarked models with 1, 2, and 3 encoder layer configurations and compared them with an attention-only baseline on three unseen airspace structures.

Our training curves indicate that increasing the number of encoder layers does not reliably improve model performance. The single-layer encoder achieved the highest returns, followed by the two-layer encoder configuration, with the three-layer configuration obtaining the lowest returns. This pattern suggests that a single encoder layer may be sufficient to learn safe separation policies with speed-only adjustments.

Two factors may explain the weaker performance of the deeper models. First, the small number of training updates may have disadvantaged the deeper models, which have a greater

number of parameters. These deeper models may require a greater number of updates to match the performance of the shallower networks. Second, speed-only actions may lead deeper models to over-optimize the dense speed reward. In contrast, smaller models appear less prone to exploiting these rewards and instead learn to avoid the more consequential conflict penalties. Taken together, these factors are consistent with the higher returns we observed for the single-layer network.

A similar trend is observed when considering the safety metrics. Across 1,000 evaluation episodes over three flight sectors, the single-layer transformer network achieved the best combined safety performance with a near-zero count of near mid-air collisions (NMACs) and the lowest average time in loss of separation (LoS) among the tested configurations. It was most reliable in the structured airspaces of Case (a) and Case (b), where it also maintained close adherence to desired speed. However, performance declined slightly in the unstructured airspace of Case (c), which is consistent with the limitations of a speed-only action space for separation assurance. The two-layer encoder configurations yielded similar safety outcomes but slightly longer LoS times. The three-layer configuration performed the worst, with the highest occurrence

of NMACs.

Finally, we showed that our transformer architecture outperformed an attention-only baseline. Specifically, the 1-layer encoder configuration achieved lower NMAC counts, lower LoS time, and closer adherence to desired speeds than the baseline. This result may stem from the additional use of multi-head attention and the use of a [CLS] token conditioned on ownship information. However, this hypothesis may require further investigation.

VIII. CONCLUSION

This study demonstrates that a multi-agent reinforcement learning framework, supported by a carefully designed relative state representation and a tailored reward function, can resolve aircraft conflicts while maintaining aircraft close to their desired cruising speeds. Our evaluation of encoder depths revealed that a transformer architecture with a 1-layer encoder outperformed the deeper configurations, achieving a near-zero NMAC rate and shorter time in loss of separation. Additionally, we demonstrated that this configuration outperforms a baseline model constructed solely with pure attention. Future work could enhance the effectiveness of our methodology through refined training strategies, such as increasing the number of intersecting routes, allowing for heading or altitude changes, or modifying the reward function. Overall, our approach represents a promising step toward enhanced MARL formulation and neural network architectures for aircraft separation assurance in future high-density AAM operations.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under Grant No. 2312092. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

- [1] D.-T. Pham, N. P. Tran, S. Alam, V. Duong, and D. Delahaye, "A machine learning approach for conflict resolution in dense traffic scenarios with uncertainties," in *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar*, 2019.
- [2] M. Brittain and P. Wei, "Scalable autonomous separation assurance with heterogeneous multi-agent reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2837–2848, 2022.
- [3] D. Groot, J. Ellerbroek, and J. Hoekstra, "Comparing attention-based methods with long short-term memory for state encoding in reinforcement learning-based separation management," *Engineering Applications of Artificial Intelligence*, vol. 159, p. 111 592, 2025.
- [4] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [6] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.
- [8] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.
- [10] M. Brittain and P. Wei, "Autonomous separation assurance in an high-density en route sector: A deep multi-agent reinforcement learning approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3256–3262.
- [11] D. J. Groot, J. Ellerbroek, and J. M. Hoekstra, "Using relative state transformer models for multi-agent reinforcement learning in air traffic control," in *SESAR Innovation Days*, Seville, Spain, Nov. 2022.
- [12] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.
- [13] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [15] J. M. Hoekstra and J. Ellerbroek, "BlueSky ATC simulator project: An open data and open source approach," in *Proceedings of the 7th international conference on research in air transportation*, FAA/Eurocontrol Washington, DC, USA, vol. 131, 2016, p. 132.
- [16] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

