COMPUTING AND ALGORITHM INSIGHT
Volume 1, Issue 2, December 2023 | pp. 141-162
https://doi.org/10.63208/21018-101

OPEN ACCESS
Check for updates

LSTM-BASED DIGITAL ASSISTANCE SYSTEM FOR ENHANCED SITUATIONAL AWARENESS AND CONFLICT DETECTION IN NEXT-GENERATION AIR TRAFFIC MANAGEMENT

Zhou Yi Fan[1] ✉, Luo Jia Hui[1], and Dang Minh Khoi[2]

School of Mathematics and Statistics, Guizhou University, Guiyang, China[1]; Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam[2]

ABSTRACT

Modern air traffic management (ATM) systems rely on human-centric coordination between controllers and pilots, a framework currently challenged by increasing flight densities and diverse aircraft types in European airspace. This saturation results in heightened congestion, safety risks, and environmental impact. To address these limitations, this paper proposes a transition toward a "next-generation" ATM through enhanced airspace abstraction and situational awareness. This study contributes an analysis of contemporary ATM complexity and introduces a digital assistance system for conflict detection. Utilizing Long Short-Term Memory (LSTM) networks, the system evaluates temporal dynamics in surveillance data to classify error patterns. Training was conducted on large-scale models comprising thousands of flights, utilizing a 20–10–1 architecture with leaky ReLU and sigmoid activations, optimized via the ADAM algorithm and binary cross-entropy loss. Numerical findings demonstrate the efficacy of LSTMs in predicting critical disruptions, including adverse weather, cyber-attacks, and human-factor-related emergencies.

Keywords: *Air Traffic Management (ATM); Long Short-Term Memory (LSTM); Conflict Detection; Situational Awareness; Airspace Abstraction.*

CORRESPONDING AUTHOR:
Zhou Yi Fan
School of Mathematics and Statistics, Guizhou University, Guiyang, China
contact: *zhouyifan@gzu.edu.cn*

ARTICLE HISTORY
Received : July 5, 2023
Final Revised : October 30, 2023
Accepted : November 21, 2023
Published : December 30, 2023

© 2023 The Author(s) | Published by Integra Academic Press | This article is an open access article and distributed under the terms and conditions of the Creative Commons Attribution (CC BY-SA) license.

1. | INTRODUCTION

From the historic pioneering flight of the Wright brothers in 1903 to the pre-pandemic peak of 37,200 daily flights in 2019 [1], aviation has undergone a massive expansion. Although the COVID-19 pandemic caused a temporary cessation of activity [2,3], flight volumes are projected to recover steadily in the medium term [3], accompanied by an influx of specialized operations (SPOs) and unmanned aircraft systems (UASs). This increasing density and variety of traffic are pushing the current human-centric air traffic management (ATM) system encompassing air traffic services (ATS), flow management (ATFM), and airspace management (ASM) to its operational limits [4].

Air traffic controllers currently face a proportional increase in data, leading to information overload and stress that can impair cognitive performance and critical decision-making. Furthermore, the fragmentation of European airspace along national borders complicates surveillance, which relies on primary (PSR) [5] and secondary (SSR) [6] radar, alongside voice-based tactical interventions via VHF/HF radio. This complex, interdependent human-centered design contributes to congestion, safety risks, and environmental inefficiencies, necessitating a transition toward "next-generation" ATM systems that utilize advanced airspace abstraction and human-integrated interfaces to bolster situational awareness [7-10].

This research introduces an Augmented ATC (AATC) system designed to provide digital assistance for conflict detection by analyzing aircraft surveillance data. To evaluate the capability of monitoring temporal dynamics and classifying error patterns, the study utilizes Long Short-Term Memory (LSTM) networks—a specialized recurrent neural network (RNN) architecture that effectively addresses the vanishing gradient problem in sequential datasets [11-14].

To train these networks, large-scale air traffic models were generated using a previously established simulation framework [15], incorporating thousands of flights with various pseudo-randomly weighted conflicts. The datasets included specific percentages of manipulated flights to simulate realistic threats: weather-related events (34.26%), cyberattacks such as jamming (27.99%), spoofing (28.10%), and data injection (32.85%), as well as emergency scenarios (36.66%) and human-factor errors like altitude and speed deviations (approx. 35%) [16,17].

The implemented LSTM networks utilize a 20–10–1 architecture, Leaky ReLU and Sigmoid activations, and the ADAM optimizer with binary cross-entropy loss. Numerical results indicate high efficacy; the system achieved approximately 99% training accuracy for cyberattacks, technical malfunctions, and human factors (loss < 0.2), and 90% for weather events (loss < 0.3). These findings, developed using Python and TensorFlow, suggest that LSTM-based temporal monitoring can significantly enhance situational awareness for air traffic controllers.

2. | RELATED WORK

To modernize and optimize European airspace, the European Commission launched the Single European Sky (SES) initiative in 2000, which subsequently established the SES ATM Research (SESAR) program. This framework produced the European ATM Master Plan, serving as a strategic roadmap for a modernized "next-generation" infrastructure. Despite widespread stakeholder support, operational progress remains sluggish. The original SES objectives—aiming for a tenfold increase in safety, a 10% reduction in environmental impact,

a 50% decrease in service costs, and a tripling of capacity by 2020—have not been fully attained. Consequently, European airspace remains fragmented by national borders, leading to persistent challenges in congestion, operational costs, and emissions [7,8].

While the human-centered design of ATM has historically ensured transport safety, the contemporary surge in traffic volume and aircraft diversity has resulted in data saturation. This information overload induces stress and potentially compromises the cognitive performance of air traffic controllers, highlighting the urgent necessity for "next-generation" systems that leverage human-integrated technologies for improved airspace abstraction and situational awareness [9,10].

Recent advancements in Artificial Intelligence (AI), specifically Deep Learning (DL), offer significant flexibility through the analysis of big data. Architectures such as Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) have been successfully deployed across diverse fields, including medical diagnostics [18], meteorological forecasting [19], seismic classification [20], spacecraft magnetic modeling [21], and traffic flow prediction [22–24]. Within the aviation sector, research into big data platforms [25], workload forecasting [26], resilient communications [27], delay estimation [28], and collaborative security management [29] underscores the transformative potential of automation. Furthermore, studies on human-computer interaction and integrated automation [30–33] emphasize the feasibility of AI in addressing systemic complexities.

To address the existing scientific gap, this research evaluates the efficacy of DL and optimization techniques as digital assistance systems for real-time conflict prediction. Given that aircraft surveillance data constitutes time-series information, this study specifically investigates the application of LSTM networks to identify emerging conflicts within synthesized air traffic datasets.

3. | BACKGROUND

Intelligence may be defined as the capacity to acquire and implement methodologies for problem-solving and goal attainment within dynamic and uncertain environments. Mirroring biological evolution, which has refined human visual recognition and adaptability, Deep Learning (DL) represents a biologically inspired programming paradigm. As a subset of Machine Learning (ML) and Artificial Intelligence (AI), DL facilitates the execution of analytical and physical tasks through automated processes that minimize the necessity for human intervention [34].

Structurally, a DL network comprises an input layer, an output layer, and a variable number of hidden layers, the latter of which defines the architectural "depth." Within these layers, the number of neuron-like units determines the "width." While conventional feedforward Artificial Neural Networks (ANNs) are optimized for independent data points [35], they are ill-suited for sequential or time-series data where temporal dependencies exist. To address this, Recurrent Neural Networks (RNNs) were developed. As illustrated in Figure 1, RNNs are engineered to maintain internal states, allowing information from preceding inputs to influence subsequent outputs.

In the RNN feedforward pass, the activation vector s_t preserves the hidden cell states and the output y at time t. Commencing with an initialized zero vector s_0, the network iterates through time steps across the hidden layers. Each individual cell processes an input x—a scalar with a d-dimensional feature vector—utilizing three temporally shared weights: W_{sx}

(associated with the recurrent layer input), W_s (associated with hidden cells), and W_{sy} (associated with the output).

Furthermore, bias vectors b_s and b_y serve as temporally shared coefficients that shift the activation functions a_1 and a_2. These activation functions introduce non-linearity, determining the activation threshold of a cell and enabling the network to resolve complex, non-linear dependencies. In this study, the Leaky Rectified Linear Unit (ReLU) and Sigmoid activation functions were selected for the model architecture, as further detailed in Section 5.

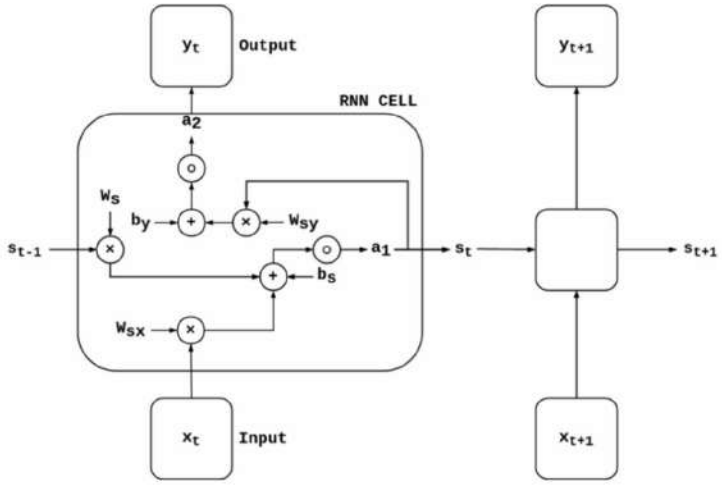


Figure 1. Structure of a RNN cell [12].

The state of a hidden cell at a specific time step t, denoted as s_t, is determined by the following expression [12]:

$$s_t = a_1(W_s s_{t-1} + W_{sx} x_t + b_s) \quad (1)$$

The computation of the hidden state involves the summation of the cross-products between the weight matrix W_s and the previous state s_{t-1}, and the weight matrix W_{sx} and the current input vector x_t, along with the bias term b_s. This sum is subsequently processed through the activation function a_1. Building upon this internal state, the network's output at time t, denoted as y_t, is derived using the following expression [12]:

$$y_t = a_2(W_{sy} s_t + b_y) \quad (2)$$

The computation involves the cross-product of the weight matrix W_{sy} and the current cell state s_t, which is subsequently summed with the bias b_y and processed via the secondary activation function a_2. While the most fundamental RNN configuration follows a one-to-one architecture where $T_x = T_y = 1$, this study implements a many-to-one architecture, as illustrated in Figure 2.

This specific configuration, characterized by $T_x > 1$ and $T_y = 1$, is conventionally utilized in applications such as sentiment classification. In the context of this research, the many-to-one structure is leveraged to map the sequence of aircraft attributes—detailed in Section 4 into a single predictive classification regarding potential air traffic conflicts. By processing a temporal window of surveillance data, the network can synthesize the historical trajectory into a final assessment of the current safety state.

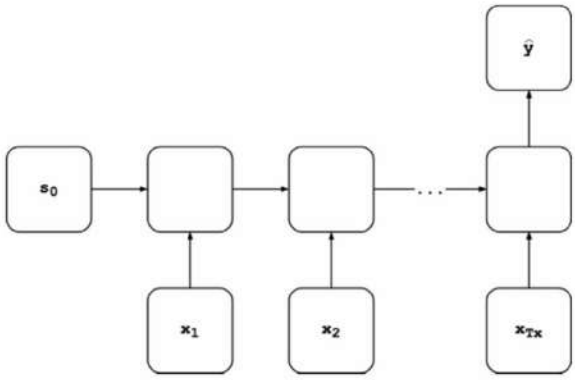


Figure 2. RNN many-to-one architecture [12].

To optimize the network, a scalar-valued metric known as the loss function L is implemented, as defined in Equation (3) [12]. This function quantifies the deviation between the network's predicted output and the ground-truth values, thereby serving as an indicator of the model's performance accuracy at each time step when processing a given dataset. The total cumulative error across all training examples is calculated as follows:

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L(\hat{y}_t, y_t) \tag{3}$$

Within the proposed AATCS, the classification task utilizes the binary cross-entropy loss function, as further elaborated in Section 5. This specific metric is designed for binary decision-making, where the output is restricted to two possible states (0 or 1). While the loss function assesses individual sequences, the cost function is applied to the entire training dataset or a specific batch during gradient descent to evaluate the overall network performance. This function quantifies the discrepancy between predicted and actual values as a singular real-valued scalar.

The primary objective of this error function is to optimize model parameters to achieve the absolute minimum error. To refine the model, an iterative process known as backpropagation is employed at each time step T, where weights and biases are updated based on the output error. The gradient of the loss function L with respect to the weight matrix W (representing the weights across all layers) is determined as follows [12]:

$$\frac{\partial L_T}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_T}{\partial W}\Big|_{(t)} \tag{4}$$

This iterative fine-tuning minimizes error rates and enhances the model's reliability by improving its capacity for generalization. The inclusion of activation functions facilitates backpropagation by providing the necessary gradients; these gradients are then utilized in conjunction with the output error to systematically update the weights and biases across the network.

4. | GENERATION AND PROCESSING OF AIR TRAFFIC DATASETS WITH AIR TRAFFIC CONFLICTS

The acquisition of high-fidelity data is fundamental for the training of AI models capable of detecting and categorizing air traffic conflicts. In this research, such conflicts are defined to include meteorological events, cyber-security threats against aviation infrastructure, emergency scenarios, and human-factor anomalies.

To generate a structured dataset for parameterized airspace abstraction, a specialized computer-based air traffic simulation and modeling framework was employed [15]. Each simulated flight comprises exactly 100 data samples, characterized by 11 specific quality attributes or features: sample number (SN), planned flight path (FP), call sign (CS), 24-bit Mode S address, squawk code, barometric altitude (ALT), indicated air speed (IAS), magnetic heading (MH), horizontal position (LAT/LON), and ATC compliance.

Diverse flight profiles were utilized to generate a user-defined volume of flights featuring pseudo-randomly weighted conflicts [15,16,17]. Meteorological disturbances (e.g., turbulence, thunderstorms, volcanic ash) were simulated by modifying waypoints and interpolating the flight path. Cyberattacks targeting Automatic Dependent Surveillance-Broadcast (ADS-B) traffic were modeled as follows:

1. Jamming: Variables such as CS, ALT, and POS are erased and replaced with "−1" over random intervals.
2. Deleting: Selective erasure of variables within specific data samples.
3. Spoofing: Manipulation of ALT, IAS, and POS coordinates.
4. Injection (Ghost Flights): Insertion of malicious ADS-B messages that simulate a legitimate flight by interpolating straight-line-on-a-sphere coordinates.

Emergency scenarios were modeled by incorporating technical malfunctions, deliberate malicious acts, and missing data across all flight phases. These scenarios utilize standardized transponder (XPDR) codes: 7500 (unlawful interference), 7600 (communications failure), and 7700 (general emergency). Human factors were simulated by inducing deviations in squawk, altitude, speed, and horizontal position resulting from pilot inattention or misinterpretation of ATC instructions. Following these manipulations, each sample was audited for ATC compliance and assigned a classification attribute.

To ensure the model generalizes effectively to unseen data and avoids overfitting (the bias-variance trade-off), a k-fold cross-validation strategy was implemented. The dataset, consisting of 3,990 flights, was randomly shuffled and partitioned into $k = 10$ independent folds. In this $k - 1$ configuration, 90% of the data is used for adjusting model weights and biases, while the remaining 10% serves as a test set for unbiased performance evaluation. This process is iterated j times to calculate the mean performance estimate E, ensuring every observation is utilized exactly once for both training and validation.

$$E = \frac{1}{10}\sum_{j=1}^{10} E_j \tag{5}$$

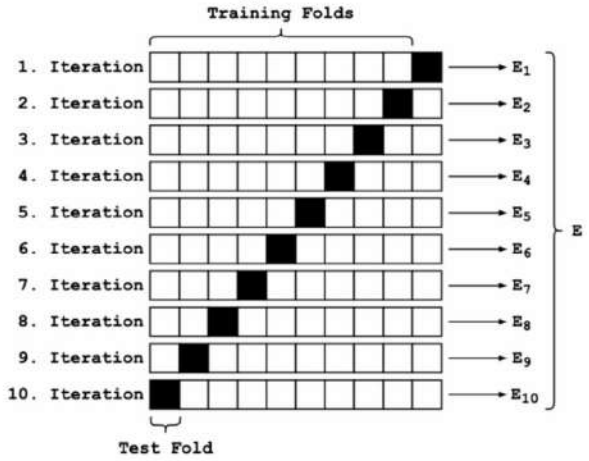


Figure 3. Cross-validation with 10 folds.

Consequently, cross-validation provides a more rigorous assessment of the model's predictive performance on novel data. However, a primary limitation of this partitioning strategy is that increasing the number of folds and iterations significantly escalates computational demand and processing time. To optimize the balance between computational expenditure and classification accuracy, a ten-fold configuration was implemented. This resulted in the generation of three-dimensional NumPy arrays: trainX $(35{,}910, 101, 11)$ and testX $(3{,}990, 101, 11)$. In these arrays, the first dimension represents the total flight count, the second signifies the 101 data samples per flight, and the third corresponds to the 11 distinctive features. Conversely, the output arrays, trainY $(35{,}910, 101, 1)$ and testY $(3{,}990, 101, 1)$, contain a singular classification attribute indicating the presence or absence of an air traffic conflict.

To ensure the efficacy of the RNN models, the input features were normalized to a uniform scale, as unscaled data typically results in substandard model performance. This study employed the MaxAbs Scaler, which is particularly suitable for datasets containing both positive and negative values. While effective, it is important to note that the MaxAbs Scaler, similar to min-max normalization, exhibits sensitivity to outliers [36,37,38]. The normalized value x_{scaled} for each feature is calculated by dividing the raw value x by the absolute maximum of that feature, resulting in a scaled range of $[-1,1]$ as defined by the following expression:

$$x_{scaled} = \frac{x}{\max(|x|)} \tag{6}$$

5. | APPLIED RECURRENT NEURAL NETWORK MODELS

This section provides a detailed overview of the RNN layer architectures, activation functions, the phenomena of vanishing and exploding gradients, gradient clipping, gating mechanisms, LSTM structures, loss functions, the ADAM optimizer, and the management of model underfitting and overfitting.

The implemented RNN (20–10–1) comprises an input layer of 20 cells, a single hidden layer containing ten cells, and an output layer with one cell. This specific architecture served as the standardized framework for all evaluated models in the subsequent analysis. Activation functions are integrated into the network to facilitate the learning of intricate data patterns by transforming the output signal of a preceding cell into a suitable input for the succeeding one. While linear activation remains the most basic form performing no transformation the Rectified Linear Unit (ReLU) has emerged as the most prevalent ramp function [12,39]. In the ReLU function, positive inputs are passed directly to the output, while negative inputs are mapped to zero, resulting in a value range of $[0, \infty]$, as illustrated in Figure 4.

The primary advantages of ReLU include an accelerated learning rate and reduced computational requirements, as it promotes sparse activation where only a subset of cells is active at any given time. This enables the network to effectively capture nonlinear dependencies and is mathematically expressed as follows [12]:

$$y = \mathrm{ReLU}(x) = x \text{ for } x > 0, \text{ else } 0 \tag{7}$$

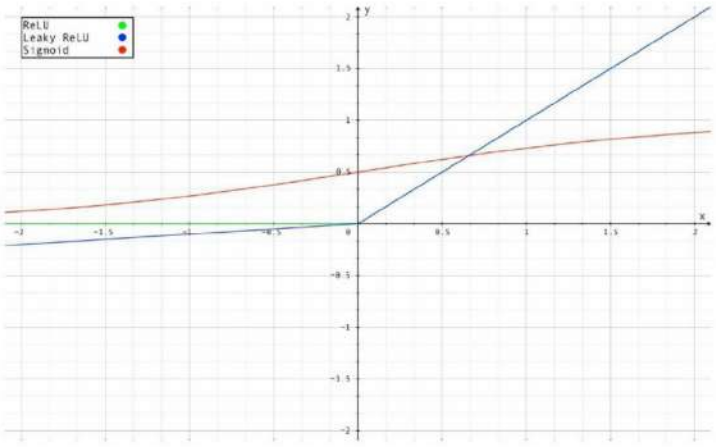


Figure 4. Activation functions.

In mathematical terms, the derivative of the ReLU function is essential for the backpropagation process, as it determines how the weights are updated. Its derivative is defined as a step function:

$$\frac{dy}{dx} = f'(x) = 1 \text{ if } x > 0, \text{ else } 0 \tag{8}$$

$$f'(x) = for x > and f'(x) = for x \tag{9}$$

In this study, the Leaky ReLU activation function is predominantly employed across the network architecture, with the specific exception of the output layer [39–41], where a Sigmoid activation function is utilized [12,39]. Leaky ReLU functions as an evolution of the standard ReLU; however, it distinguishes itself by maintaining a small, non-zero gradient even when a cell is saturated or inactive. For all evaluated RNN models in this research, the negative slope coefficient, denoted as α, is established at 0.1 (refer to Figure 4). This configuration extends the potential value range from $[-\infty, \infty]$, ensuring that the network can continue learning from negative input values.

The Leaky ReLU function is mathematically defined as follows:

$$y = \text{Leaky ReLU}(x) = f(x) = \begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases}, a > 0 \tag{10}$$

The derivative is noted as follows:

$$\frac{dy}{dx} = f'(x) = \frac{d}{dx}\left(\begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases}\right), a > 0 \tag{11}$$

$$f'(x) = \begin{cases} 1, & x > 0 \\ a, & x \leq 0 \end{cases}, a > 0 \tag{12}$$

Alternatively, the sigmoid activation function is utilized for logistic regression [12, 39, 40]. As previously established, this function is implemented in the output layer for binary classification tasks, effectively mapping input signals to a probability space between zero and one (see Figure 4). Given that the output values are bounded within the [0,1] interval, the final prediction is categorized as one if the calculated value exceeds 0.5; conversely, if the value is below this threshold, the result is classified as zero. The sigmoid activation function, formally denoted by the symbol σ, is mathematically defined as follows [12]:

$$y = \sigma(x) = f(x) = \frac{1}{1+e^{-x}} \tag{13}$$

This results in the following derivation:

$$\frac{dy}{dx} = f'(x) = f(x)[1 - f(x)] \tag{14}$$

Deep sequential neural networks, such as RNNs, frequently encounter the phenomena of vanishing and exploding gradients during the training phase [11, 12, 13, 14, 34, 35]. In this framework, the gradient represents both the direction and magnitude required for weight adjustments to minimize the output loss. However, these gradients may accumulate during updates, leading to excessively small or large values. Standard RNN models typically employ the hyperbolic tangent (tanh) activation function, which is bounded between $[-1,1]$ and possesses a derivative range of $[0,1]$. Due to the depth of the network, the number of matrix multiplications increases alongside the input sequence length. When applying the chain rule during backpropagation, the repeated multiplication of small fractional values causes the gradient to diminish exponentially toward zero. This results in the vanishing gradient problem, where weights cease to update, effectively halting the learning process. Notably, implementing alternative activation functions, such as the Leaky ReLU, does not inherently resolve the inability of standard RNNs to manage long-term dependencies.

Conversely, the exploding gradient problem operates on a reciprocal principle. When gradients with values greater than one are repeatedly multiplied across network layers, they experience exponential growth. This surge can lead to numerical overflow, resulting in "NaN" values and unstable weight updates that prevent the RNN from converging. To mitigate exploding gradients, strategies such as reducing network complexity, employing smaller batch sizes, utilizing truncated backpropagation through time (BPTT), or transitioning to an LSTM architecture are recommended.

The LSTM architecture is designed as a stack of neural networks with linear layers, comprising biases and weights that are iteratively refined via backpropagation. These networks are specifically engineered to manage long-term dependencies, noise, and continuous values while addressing gradient instability without altering the fundamental training model. If exploding gradients persist, a technique known as gradient clipping is used to cap the maximum gradient value. To overcome the vanishing gradient problem, LSTM cells (see Figure 5) incorporate a gating mechanism within the hidden layers to regulate the flow of information effectively.

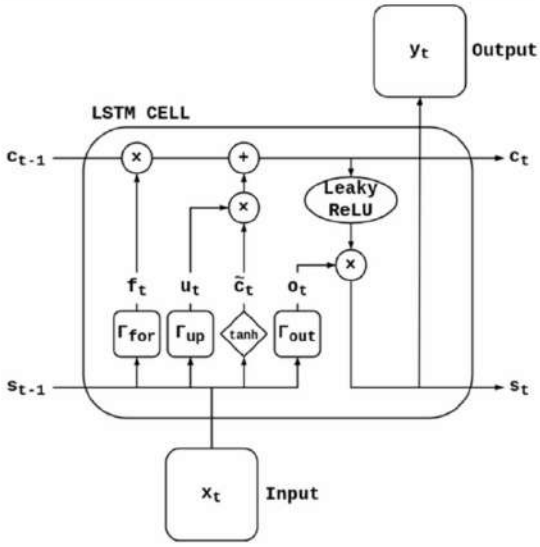


Figure 5. illustrates the internal architecture and gating mechanisms of a LSTM cell [12].

The gating mechanism facilitates the retention of values across arbitrary temporal intervals, utilizing gates designated as Γ (refer to Equation (15) [12]) to modulate the flow of information into and out of the cell. These gates preserve memory components in an analog format and generate probabilistic scores through pointwise multiplication using sigmoid activation functions. A formal distinction is maintained between the update gate Γ_{up}, the forget gate Γ_{for}, and the output gate Γ_{out}. While the update gate Γ_{up} determines the relevance of current input for inclusion, the forget gate Γ_{for} identifies and eliminates non-essential information by scaling irrelevant inputs to zero, thus enabling the forgetting process. Concurrently, the output gate Γ_{out} governs which portions of the current cell state c_t are manifested in the output and finalizes the subsequent hidden state. The underlying linear transformations are governed by matrices W_{xm} and W_{sm}, which represent the weights of the input and recurrent connections, respectively, alongside the bias vector b_m. Within this framework, the subscript m serves as a functional placeholder for the specific gate or memory cell being computed.

$$\Gamma_m = \sigma(W_{xm}x_t + W_{sm}s_{t-1} + b_m) \tag{15}$$

The architectural framework of each cell is characterized by a configuration of three inputs and two distinct outputs. The input signals comprise the current data point or input vector x_t, the antecedent cell state c_{t-1}, and the hidden state derived from the preceding cell s_{t-1}. Regarding the output interface, the system utilizes the contemporary cell state c_t alongside the revised hidden state s_t, which is formally identified as the output vector of the LSTM cell. Within this structure, the forget gate Γ_{for} processes two specific inputs, s_{t-1} and x_t, to generate a control signal f_t (refer to Equation (16)) constrained within the interval of [0, 1]. This resulting output subsequently undergoes pointwise multiplication with the previous cell state c_{t-1} (refer to Equation (20)), a critical step in the information regulation process.

$$f_t = \sigma\left(W_{xf}x_t + W_{sf}s_{t-1} + b_f\right) \tag{16}$$

Γup takes the same inputs as the Γfor and generates the output ut, calculated as follows:

$$u_t = \sigma(W_{xu}x_t + W_{su}s_{t-1} + b_u) \tag{17}$$

Γout takes the same inputs as Γfor and Γup and generates the output ot, noted as follows:

$$o_t = \sigma(W_{xo}x_t + W_{so}s_{t-1} + b_o) \tag{18}$$

The integration of new information into the cell state is facilitated by the element-wise product of the update gate u_t and the candidate cell input state vector $\widetilde{c_t}$ (refer to Equation (19)), the latter of which is generated by a hyperbolic tangent (tanh) layer to define the magnitude of information designated for storage. The resulting value is subsequently aggregated with the product of the forget gate f_t and the previous cell state c_{t-1} to derive the updated cell state c_t (refer to Equation (20)). In the final stage of the computational cycle, the LSTM output vector s_t (refer to Equation (21)) is determined through the pointwise multiplication of the output gate o_t and a leaky ReLU activation layer. This specific activation function is employed to modulate the output, shifting it into the broader range of $[-\infty, \infty]$ to complete the hidden state transition.

$$\widetilde{c_t} = \tanh(W_{xc}x_t + W_{sc}s_{t-1} + b_c) \tag{19}$$

$$c_t = f_t \otimes c_{t-1} + u_t \otimes \widetilde{c_t} \tag{20}$$

$$s_t = o_t \otimes \text{leaky ReLU}(c_t) \tag{21}$$

The interaction between the previously defined gates facilitates a synergistic process, whereby the internal cell state and the final output are generated concurrently through a series of regulated feedback loops. To optimize the training of this LSTM architecture, the ADAM (Adaptive Moment Estimation) algorithm [42, 43, 44] is employed as the primary gradient descent optimization technique. This optimizer functions by integrating the principles of gradient descent with momentum and Root Mean Square Propagation (RMSProp), allowing for adaptive learning rates. The selection of ADAM is justified by its computational efficiency, minimal memory requirements, and robust performance when processing high-dimensional datasets with extensive feature sets. The governing dynamics of the ADAM optimizer are formulated as follows:

$$W_{t+1} = W_t - \widehat{m_t}\left(\frac{\alpha}{\sqrt{\widehat{v_t}}+\epsilon}\right) \tag{22}$$

In this formulation, $\widehat{m_t}$ and $\widehat{v_t}$ (refer to Equation (22) [42, 43, 44]) represent the bias-corrected weight parameters, while α denotes the learning rate, assigned as a specific floating-point value for each trained RNN (refer to Section 6). This rate dictates the velocity at which the network adjusts its weights and biases during each iteration. For the exponential decay rates associated with the moment estimates, the configuration utilizes β_1 (momentum) and β_2 (RMSP), alongside the constant ϵ for numerical stability; these are set to the default Keras values of 0.9, 0.999, and 1×10^{-7}, respectively.

To facilitate the learning process of the implemented RNN models, the binary cross-entropy loss function is employed [45, 46, 47, 48]. This objective function involves computing the logarithms of $\widehat{y_t}$, representing the t-th scalar output value, and $(1 - \widehat{y_t})$ within the bounded interval of $(0,1)$. The total number of scalar values in the RNN output is denoted by N, while the corresponding target value is defined as y_t (refer to Equation (23) [45, 46, 47, 48]).

$$L = -\frac{1}{N}\sum_{t=1}^{N}[y_t \cdot \log(\widehat{y_t}) + (1 - y_t) \cdot \log(1 - \widehat{y_t})] \tag{23}$$

It is important to observe that the binary cross-entropy loss function is specifically compatible with the sigmoid activation function in the output layer, as the latter ensures that output values are constrained within the $(0,1)$ interval.

The primary objective of Artificial Neural Network (ANN) and Recurrent Neural Network (RNN) models is to achieve an optimal balance between learning the training data and generalizing to unseen datasets, a concept known as the bias–variance trade-off. Overfitting occurs when a network learns the training data excessively well, resulting in low bias but high variance, which leads to performance fluctuations when encountering new data. Conversely, underfitting is characterized by a high bias and low variance, occurring when the network fails to sufficiently capture the underlying patterns of the problem, irrespective of the training samples.

To mitigate underfitting, it is often necessary to increase network complexity, extend the number of training epochs, or expand the feature set while filtering statistical noise from the input. To prevent overfitting, strategies include increasing the volume of training data and reducing network complexity by modifying its structure and weight parameters. Regularization methods are specifically employed to minimize generalization errors by maintaining smaller weight magnitudes. Common techniques include:

1. Weight Constraints: Restricting the size of weights to a specific range or threshold.

2. Dropout: The stochastic removal of input units during the training phase, where each cell is retained based on a predefined probability.
3. Noise Injection: The deliberate addition of statistical noise to inputs during training to improve robustness.
4. Early Stopping: Monitoring the model's performance and terminating the training process as soon as the validation loss begins to diverge or fail to decrease.
5. Give your comment here as a closing statement. This can be the final conclusion of your discussion and analysis and your recommendations for further research projects. Conclusions can include the main points of this paper, but do not replicate abstracts in conclusions.

This section not only shows the main shortcomings and limitations of studies that can reduce the validity of writing, thus raising the question of the reader (whether, or in what way) the limits in his study might have influenced the results and conclusions. The author can provide answers to possible questions that arise, for example: Is this a problem caused by an error, or in the method chosen, or its validity, or vice versa?

6. | ANALYSIS OF THE APPLIED RECURRENT NEURAL NETWORK MODELS

To evaluate model performance and facilitate intuitive interpretation, accuracy is employed as a primary metric, monitoring the alignment between predicted and ground-truth values throughout the training phase. Additionally, the loss value serves as a critical indicator of the network's behavior following each optimization iteration, as detailed in Section 3.

This section evaluates the efficacy of the models in predicting air traffic conflicts, specifically focusing on weather events. The training dataset consists of 3,990 flights, of which 34.26% have been manipulated to include weather-related conflict events. Consistent with the architecture described in Section 5, the network utilizes a 20–10–1 layer configuration, employing leaky ReLU and sigmoid activation functions, the latter being situated in the output layer. The learning process is governed by a binary cross-entropy loss function and the ADAM optimizer, with a learning rate of 3×10^{-4} applied over ten epochs and a batch size of 64. In this context, the batch size represents the subset of the training data utilized in each iteration, selected via random sampling.

The application of this methodology resulted in a final training accuracy of 90.24% with a corresponding loss of 0.2448. Notably, the validation phase yielded slightly superior performance, achieving an accuracy of 90.29% and a loss value of 0.2442. Regarding the longitudinal performance of the RNN, accuracy trajectories (refer to Figure 6) demonstrate a consistent upward trend across successive epochs, while the loss trajectories (refer to Figure 7) asymptotically approach zero.

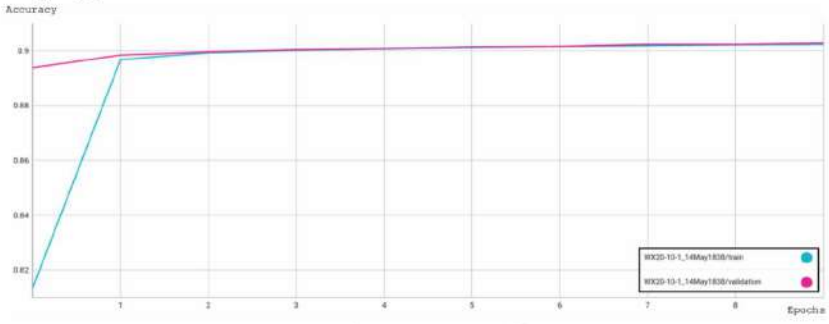


Figure 6. Weather events: model accuracy.

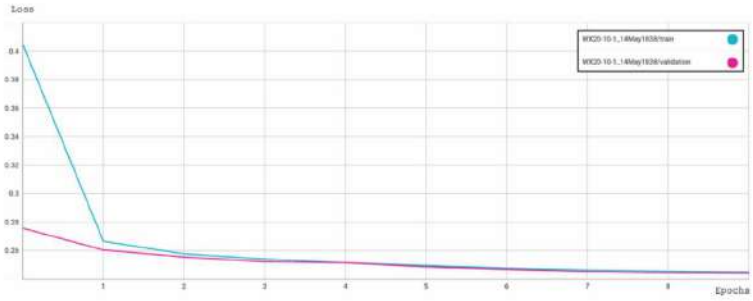


Figure 7. Weather events: model loss.

Figure 8 presents the histogram plots illustrating the distribution of the bias, kernel, and recurrent kernel parameters for LSTM cell three. Within these visualizations, the x-axis delineates the range of possible weight and bias values for the RNN, while the y-axis denotes the frequency of these observed values. To analyze the temporal progression of these parameters, the histograms for individual epochs are superimposed. This layered approach utilizes color intensity to represent training duration: lighter histograms correspond to earlier epochs, whereas darker shades signify the parameter distributions after several training cycles.

An analysis of these distributions reveals specific training dynamics; a relatively homogeneous, single-peaked distribution suggests a fine-tuning process concentrated around a specific parameter value. Conversely, the presence of multiple peaks or "bumps" indicates that parameter updates are oscillating or shifting between multiple distinct values throughout the training phase.

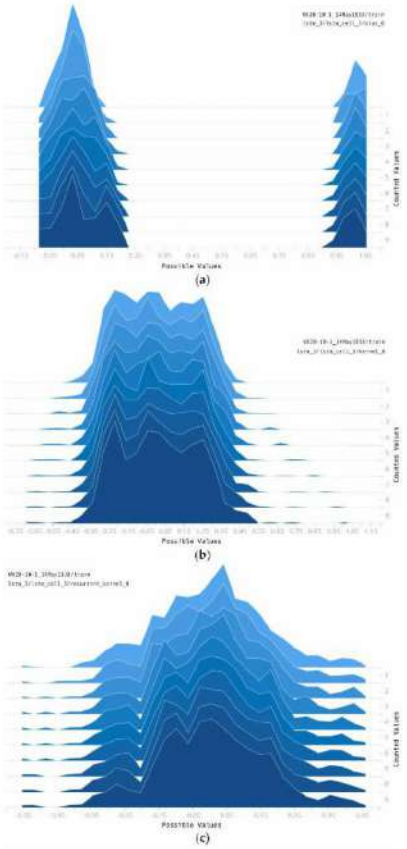


Figure 8. Histogram distributions for weather event prediction, depicting the parameter weights of the third LSTM cell: (a) Bias vector distribution; (b) Input kernel weights; (c) Recurrent kernel weights.

An alternative perspective on these parameters is provided by the distribution plots for the bias, kernel, and recurrent kernel of the third LSTM cell (refer to Figure 9). In this visualization, the x-axis represents the progression of epochs, while the y-axis delineates the range of possible parameter values. To represent the frequency of specific values during the training phase, the plot utilizes varying levels of color saturation; higher concentrations of occurrences are signified by darker data points, which fade progressively as the frequency decreases. The efficacy of the model's evolution is evaluated by the homogeneity of the gradient along the y-axis as training advances.

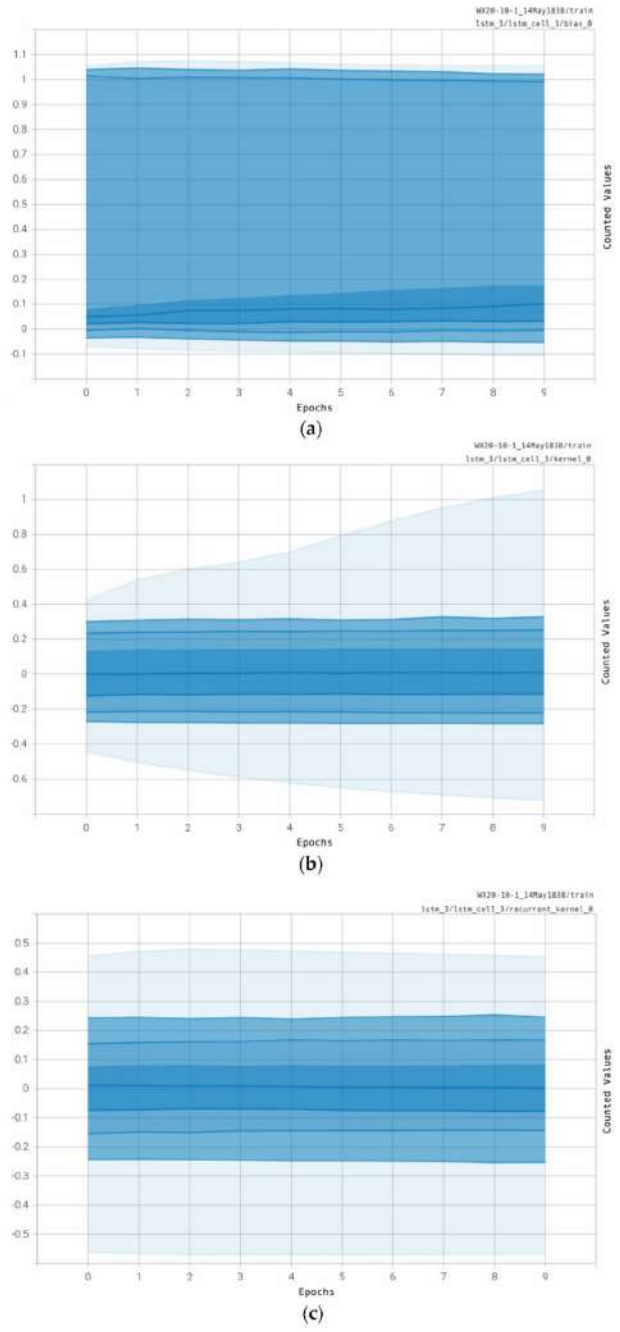


Figure 9. Distributional trends in weather event forecasting for LSTM cell three, showing the progression of (a) biases, (b) kernels, and (c) recurrent kernels across training epochs.

The subsequent analysis focuses on cybersecurity threats, specifically investigating jamming, deleting, spoofing, and injecting (ghost flight) attacks. Each of the four datasets comprises 3,990 flights, with manipulation rates of 27.99% for jamming, 27.98% for deleting, 28.10% for spoofing, and 32.85% for injecting. While the input-output array configurations and the general RNN architecture remain consistent with previous experiments, the hyperparameters for the ADAM optimizer and batch sizes were adjusted to suit each specific attack vector. Specifically, the learning rates were established at 6×10^{-3} for jamming, 2×10^{-4} for deleting, and 1×10^{-4} for both spoofing and injecting. Batch sizes were set to 128 for the first three categories and 64 for the injecting dataset.

The implementation of these parameters yielded high performance across all categories. Jamming achieved a training accuracy of 99.79% (loss: 4.3×10^{-3}) and a validation accuracy of 99.79% (loss: 3.8×10^{-3}). The deleting attack reached a training accuracy of 99.63% (loss: 7×10^{-3}) and a validation accuracy of 99.67% (loss: 6.4×10^{-3}). Spoofing resulted in a training accuracy of 98.98% (loss: 3.62×10^{-2}) and a validation accuracy of 99.30% (loss: 2.59×10^{-2}). Finally, the injecting attack recorded a training accuracy of 99.78% (loss: 1.85×10^{-2}) and a validation accuracy of 99.90% (loss: 1.61×10^{-2}).

As illustrated in the accuracy and loss trajectories (refer to Figures 10 and 11), the models demonstrated a consistent convergence pattern where accuracy increased and loss approached zero over the training duration. However, the learning and attenuation rates varied significantly between attack types. Notably, the loss function for the deleting attack exhibited an upward shift between epochs eight and nine, suggesting the onset of overfitting. This fluctuation indicates that terminating the training process at this juncture may be necessary to maintain optimal validation performance and prevent the model from losing its generalizability.

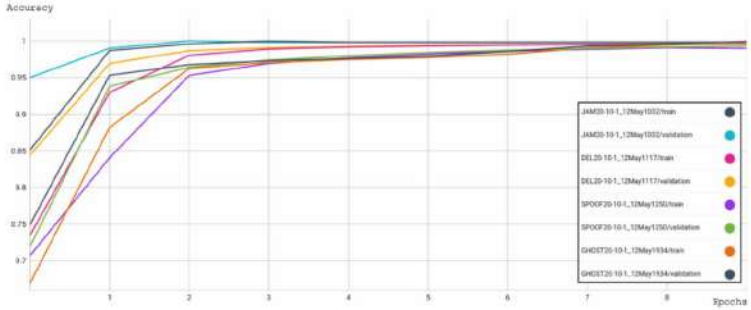


Figure 10. Cyberattacks: model accuracy.

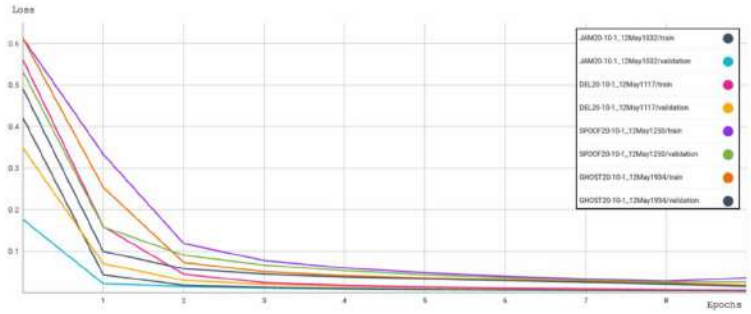


Figure 11. Cyberattacks: model loss.

The analysis concludes with an evaluation of aeronautical emergency conditions, specifically examining the three critical XPDR emergency codes: 7700 (General Emergency),

7600 (Radio Failure), and 7500 (Unlawful Interference). Consistent with previous experiments, each dataset comprises 3,990 flights, with manipulation frequencies of 36.66% for code 7700, 36.64% for 7600, and 34.98% for 7500. While the fundamental RNN architecture, epoch count, and input-output array structures are maintained, the ADAM optimizer hyperparameters were customized for each scenario. The learning rates were calibrated to 2×10^{-3} for code 7700, 1×10^{-4} for 7600, and 9×10^{-5} for 7500, with a uniform batch size of 128 applied across all three datasets.

The experimental results demonstrate high predictive precision for emergency detection. For the 7700 dataset, the model achieved a training accuracy of 99.28% (loss: 1.68×10^{-2}) and a validation accuracy of 99.31% (loss: 1.58×10^{-2}). The 7600 scenario yielded a training accuracy of 99.58% (loss: 1.2×10^{-2}) and a validation accuracy of 99.62% (loss: 1.1×10^{-2}), while the 7500 dataset reached 99.54% training accuracy (loss: 3.95×10^{-2}) and 99.65% validation accuracy (loss: 3.39×10^{-2}).

Consistent with the patterns observed in prior classifications, the training and validation metrics remain closely aligned, indicating strong model stability. As illustrated in the performance trajectories (refer to Figures 12 and 13), accuracy exhibits a distinct upward trend across epochs as the loss function asymptotically approaches zero. Each emergency type displays unique learning velocities and loss attenuation rates, reflecting the distinct data characteristics of each transponder code.

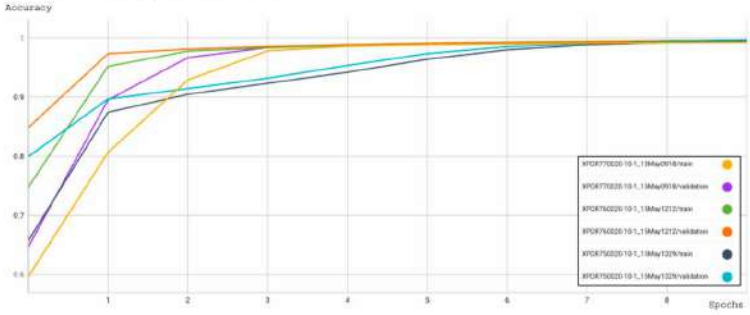


Figure 12. Emergency situations: model accuracy.

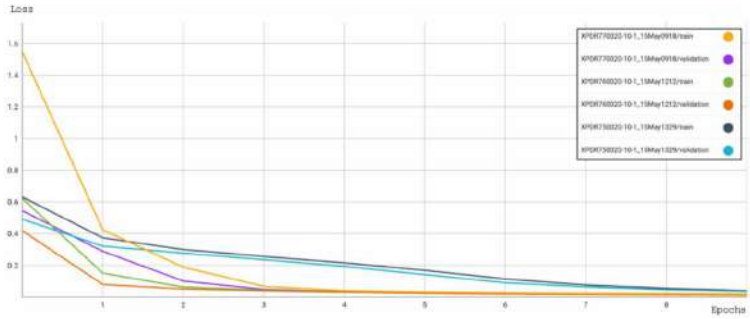


Figure 13. Emergency situations: model loss.

Lastly, the investigation evaluates human factors resulting from pilot inattention, incorrect actions, or non-executed instructions. Specifically, the analysis examines incorrect transponder (squawk) settings as well as horizontal position (POS), altitude (ALT), and speed (SPD) deviations. Consistent with the preceding experimental sets, each dataset comprises 3,990 flights. The proportion of manipulated flights within these datasets is 36.20% for incorrect squawk, 35.53% for horizontal POS, and 35.36% for both ALT and SPD. The structural

configuration of the input and output arrays remains consistent, with the output arrays indicating the presence or absence of a human factor discrepancy. The fundamental RNN architecture and the designated number of epochs also remain unchanged.

However, adjustments were made to the ADAM optimizer's learning rates and the batch sizes. The learning rates were established at 2×10^{-4} for both incorrect squawk and ALT deviations, and 1×10^{-4} for horizontal POS and SPD deviations. A uniform batch size of 128 was applied across all four datasets. These parameters yielded a final training accuracy of 99.80% (loss: 1.22×10^{-2}) for incorrect squawk, with a corresponding validation accuracy of 99.89% (loss: 1.04×10^{-2}). Horizontal POS deviations reached a training accuracy of 99.68% (loss: 1.03×10^{-2}) and a validation accuracy of 99.69% (loss: 9.6×10^{-3}). For ALT deviations, the model achieved a training accuracy of 99.82% (loss: 5.9×10^{-3}) and a validation accuracy of 99.83% (loss: 5.3×10^{-3}). Finally, SPD deviations resulted in a training accuracy of 99.57% (loss: 1.52×10^{-2}) and a validation accuracy of 99.59% (loss: 1.42×10^{-2}).

As observed in the previous categories, the training and validation outcomes are nearly identical, suggesting high model stability. The accuracy trajectories (refer to Figure 14) exhibit a distinct upward trend across epochs, while the loss trajectories (refer to Figure 15) asymptotically approach zero. Furthermore, it is evident that each human factor category exhibits unique learning velocities and distinct rates of loss attenuation.

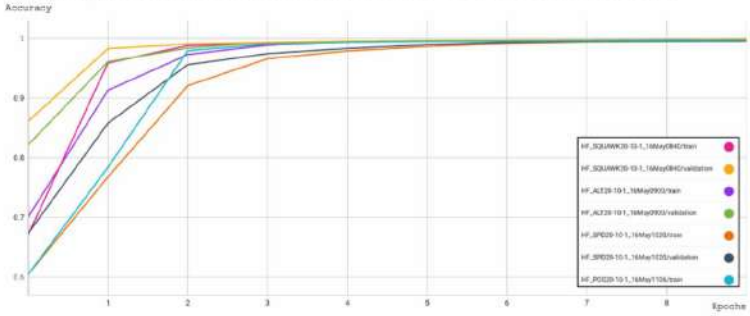


Figure 14. Human factors: model accuracy.

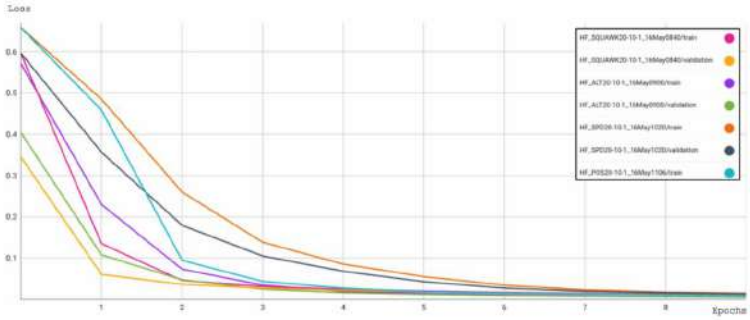


Figure 15. Human factors: model loss.

A comparative analysis of the overall accuracies achieved across all test cases is illustrated in Figure 16. The data indicates that the model consistently exceeds a 99% accuracy threshold for all scenarios with the singular exception of weather influence management. In the latter case, the accuracy remains above the 90% mark. This performance disparity may be attributed to the nature of weather-related maneuvers, which often involve only marginal adjustments to

the template flight trajectory. Consequently, such subtle deviations may lack the distinct signatures present in other conflict categories, leading to a comparatively lower, yet still robust, predictive performance.

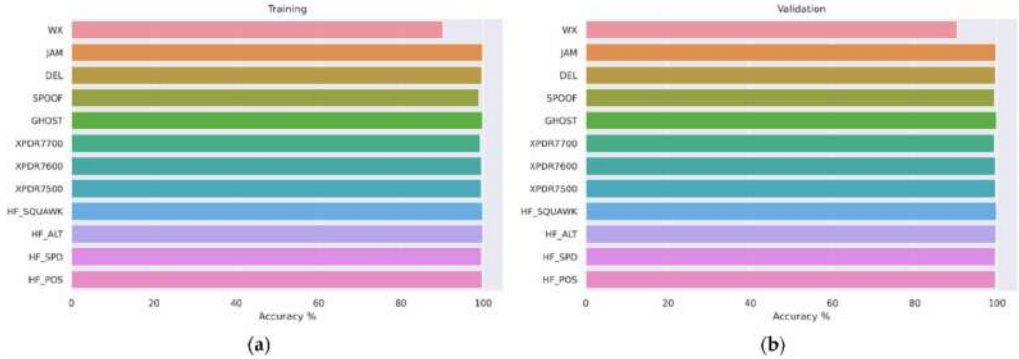


Figure 16. Comparative accuracy of the RNN models across various conflict scenarios, including weather (WX), cyberattacks (JAM, DEL, SPOOF, GHOST), emergency codes (XPDR), and human factors (HF): (a) training performance and (b) validation performance for all evaluated models.

7. | CONCLUSIONS

Historical advancements in Air Traffic Management (ATM) have established a high standard of aviation safety; however, the contemporary surge in flight volume and the diversification of aircraft types in European airspace are straining the traditional human-centric paradigm. Furthermore, the integration of unmanned and self-piloted operations, coupled with the fragmentation of airspace along national boundaries, has created significant utilization bottlenecks. Although initiatives like the Single European Sky (SES) aim to modernize and reorganize these structures, progress remains sluggish [7], primarily due to political complexities rather than technological limitations within the SESAR ATM master plan [8].

Under these prevailing conditions, maintaining a human-centered design contributes to critical issues, including congestion, diminished safety margins, increased operational costs, delays, and elevated emissions. Consequently, there is an urgent requirement to transition toward "next-generation" ATM systems. Emerging Communications, Navigation, and Surveillance (CNS) technologies—such as ADS-B, Controller Pilot Data Link Communications (CPDLC), and the developing L-Band Digital Aeronautical Communication System (LDACS)—offer significant potential for acquiring high-fidelity aircraft data. By integrating these real-world data streams with simulated models of realistic air traffic conflicts, a parameterized abstraction of the airspace can be achieved.

This research focuses on the development of mechanisms for the detection and resolution of air traffic conflicts through a digital assistance system powered by Artificial Intelligence (AI). By systematically analyzing aircraft surveillance data, the proposed system aims to reduce air traffic controller workload and enhance situational awareness by providing early alerts regarding emerging conflicts. To achieve this, Deep Learning (DL) techniques are employed. The temporal dynamic behavior of the LSTM networks utilized in this study demonstrates robust capabilities in monitoring, classifying, and predicting conflicts within sequential flight data. These capabilities are empirically supported by the model's performance metrics, where accuracy consistently improved and loss values asymptotically approached zero within a few training epochs.

Furthermore, this study highlights that there is no universal "golden rule" for establishing the optimal architecture for Recurrent Neural Network (RNN) models. Determining the ideal configuration of hyperparameters—including the number of hidden layers, cell density, activation and loss functions, weight initialization, dropout, and optimizer settings (such as learning rate and batch normalization)—remains a complex and time-intensive process. In this research, approximately 100 distinct combinations were evaluated to achieve an architecture where errors were sufficiently minimized. Beyond architectural challenges, the generation of large-scale, realistic air traffic models involving thousands of flights was essential to expose the LSTM networks to diverse scenarios, allowing them to identify patterns by adjusting internal weights and biases.

Ultimately, this paper underscores the transformative potential of AI in human-integrated systems and highly automated ATM solutions. Such technologies are vital for future fully Unmanned Traffic Management (UTM) networks, promising to enhance human performance and flight safety while simultaneously reducing costs, operational delays, and environmental emissions.

REFERENCES

1. EUROCONTROL. Daily Traffic Variation. Available online: https://www.eurocontrol.int/Economics/2020-DailyTrafficVariation-States.html (accessed on 25 June 2022).
2. EUROCONTROL. Forecast Update 2021–2024. Available online: https://www.eurocontrol.int/sites/default/files/2021-05/eurocontrol-four-year-forecast-2021-2024-full-report.pdf (accessed on 28 October 2021).
3. EUROCONTROL. 3-Year Forecast 2022–2024. Available online: https://www.eurocontrol.int/news/eurocontrol-3-year-forecast-2022–2024 (accessed on 25 June 2022).
4. SKYBRARY Aviation Safety. Air Traffic Management (ATM). Available online: https://skybrary.aero/articles/air-traffic-management-atm (accessed on 5 April 2022).
5. SKYBRARY Aviation Safety. Primary Surveillance Radar (PSR). Available online: https://skybrary.aero/articles/primary-surveillance-radar-psr (accessed on 5 April 2022).
6. SKYBRARY Aviation Safety. Secondary Surveillance Radar (SSR). Available online: https://skybrary.aero/articles/secondary-surveillance-radar-ssr (accessed on 5 April 2022).
7. International Air Transport Association. Single European Sky. Available online: https://www.iata.org/en/about/worldwide/europe/single-european-sky/ (accessed on 5 March 2022).
8. European ATM Master Plan. Available online: https://www.atmmasterplan.eu/downloads/285 (accessed on 5 March 2022).
9. Landry, S.J. Human centered design in the air traffic control system. J. Intell. Manuf. 2011, 22, 65–72.
10. Zhao, Y.; Chen, K. Air traffic congestion assessment method based on evidence theory. In Proceedings of the 2010 Chinese Control and Decision Conference, Xuzhou, China, 26–28 May 2010; pp. 426–429.

11. Li, F.-F.; Johnson, J.; Young, S. Recurrent Neural Networks. Available online: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf (accessed on 23 March 2022).
12. Amidi, A.; Amidi, S. Recurrent Neural Networks Cheatsheet. Available online: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks (accessed on 23 March 2022).
13. Saeed, M. An Introduction to Recurrent Neural Networks and the Math that Powers Them. Available online: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/ (accessed on 23 March 2022).
14. Hochreiter, S.; Schmiedhuber, J. Long Short-Term Memory. Available online: http://www.bioinf.jku.at/publications/older/2604.pdf (accessed on 23 March 2022).
15. Ortner, P.; Steinhöfler, R.; Leitgeb, E.; Flühr, H. Air Traffic Simulation and Modeling Framework. In Proceedings of the 2022 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom), Graz, Austria, 12–14 July 2022.
16. Ortner, P.; Flühr, H.; Leitgeb, E. Cyber Security and Information Exchange Analysis of Automatic Dependent Surveillance Broadcast. In Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 19–21 September 2019; pp. 1–6.
17. Ortner, P.; Steinhöfler, R.; Hödl, P.; Leitgeb, E.; Flühr, H. Geospatial Guidance of Unmanned Aerial Vehicles around no-fly-zones by Global Positioning System Spoofing. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 1–3 June 2021; pp. 1–7.
18. Kumar, G.; Kumar, P.; Kumar, D. Brain Tumor Detection Using Convolutional Neural Network. In Proceedings of the 2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, India, 3–4 December 2021; pp. 1–6.
19. Llugsi, R.; Yacoubi, S.E.; Fontaine, A.; Lupera, P. Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito. In Proceedings of the 2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM), Virtual Event, 24–26 November 2021; pp. 1–6.
20. Mustika, I.W.; Adi, H.N.; Najib, F. Comparison of Keras Optimizers for Earthquake Signal Classification Based on Deep Neural Networks. In Proceedings of the 2021 4th International Conference on Information and Communications Technology (ICOIACT), Virtual Event, 30–31 August 2021; pp. 304–308.
21. Spantideas, S.T.; Giannopoulos, A.E.; Kapsalis, N.C.; Capsalis, C.N. A Deep Learning Method for Modeling the Magnetic Signature of Spacecraft Equipment Using Multiple Magnetic Dipoles. IEEE Magn. Lett. 2021, 12, 1–5.
22. Wang, Z.; Zhu, R.; Zheng, M.; Jia, X.; Wang, R.; Li, T. A Regularized LSTM Network for Short-Term Traffic Flow Prediction. In Proceedings of the 2019 6th International Conference on Information Science and Control Engineering (ICISCE), Shanghai, China, 20–22 December 2019; pp. 100–105.
23. Chang, Z.; Zhang, Y.; Chen, W. Effective Adam-Optimized LSTM Neural Network for Electricity Price Forecasting. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018; pp. 245–248.

24. Wei, C.; Qi, L. Weak Signal Detection Based on WT-LSTM. In Proceedings of the 2020 IEEE 3rd International Conference on Electronic Information and Communication Technology (ICEICT), Shenzhen, China, 13–15 November 2020; pp. 393–397.
25. Chen, Y.; Zhou, L.; Yang, J.; Yan, Y. Big Data Platform of Air Traffic Management. In Proceedings of the 2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT), Kunming, China, 17–19 October 2019; pp. 137–141.
26. Wang, H.; Gong, D.; Wen, R. Air traffic controllers workload forecasting method based on neural network. In Proceedings of the 27th Chinese Control and Decision Conference (2015 CCDC), Qingdao, China, 23–25 May 2015; pp. 2460–2463.
27. Kabashkin, I. Resilient communication network of Air Traffic Management system. In Proceedings of the 2016 Advances in Wireless and Optical Communications (RTUWO), Riga, Latvia, 3–4 November 2016; pp. 156–160.
28. Boggavarapu, R.; Agarwal, P.; Kumar, D.H.R. Aviation Delay Estimation using Deep Learning. In Proceedings of the 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 21–22 November 2019; pp. 689–693.
29. Hawley, M.; Howard, P.; Koelle, R.; Saxton, P. Collaborative Security Management: Developing Ideas in Security Management for Air Traffic Control. In Proceedings of the 2013 International Conference on Availability, Reliability and Security, Regensburg, Germany, 2–6 September 2013; pp. 802–806.
30. Cheng, T.; Wen, P.; Li, Y. Research Status of Artificial Neural Network and Its Application Assumption in Aviation. In Proceedings of the 2016 12th International Conference on Computational Intelligence and Security (CIS), Wuxi, China, 16–19 December 2016; pp. 407–410.
31. Lee, P.U. Understanding human-human collaboration to guide human-computer interaction design in air traffic control. In Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 10–12 October 2005; Volume 2, pp. 1598–1603.
32. Prevot, T. On the design of integrated air/ground automation. In Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 10–12 October 2005; Volume 4, pp. 3933–3938. [Green Version]
33. Hopkin, V.D. Safety and human error in automated air traffic control. In Proceedings of the 1999 International Conference on Human Interfaces in Control Rooms, Cockpits and Command Centres, Bath, UK, 21–23 June 1999; pp. 113–118.
34. Kang, N. Introduction Deep Learning and Neural Networks. Available online: https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883?gi=2499d405d991 (accessed on 4 February 2022).
35. Gupta, T. Deep Learning: Feed Forward Neural Network. Available online: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7 (accessed on 4 February 2022).
36. Loukas, S. Everything You Need to Know about Min-Max Normalization. Available online: https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79?gi=19aae5625a9f (accessed on 6 February 2022).
37. Brownlee, J. How to Use Data Scaling Improve Deep Learning Model Stability and Performance. Available online: https://machinelearningmastery.com/how-to-improve-

neural-network-stability-and-modeling-performance-with-data-scaling/ (accessed on 6 February 2022).
38. Scikit-Learn. MaxAbs Scaler. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.html (accessed on 6 February 2022).
39. Brownlee, J. How to Choose an Activation Function for Deep Learning. Available online: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (accessed on 10 February 2022).
40. Keras. Layer Activation Functions. Available online: https://keras.io/api/layers/activations/ (accessed on 10 February 2022).
41. Keras. LeakyReLU Layer. Available online: https://keras.io/api/layers/activation_layers/leaky_relu/ (accessed on 10 February 2022).
42. Kingma, D.P.; Ba, J.L. ADAM: A Method for Stochastic Optimization. Available online: https://arxiv.org/pdf/1412.6980.pdf (accessed on 12 February 2022).
43. Keras. Adam. Available online: https://keras.io/api/optimizers/adam/ (accessed on 12 February 2022).
44. Bushaev, V. Adam—Latest Trends in Deep Learning Optimization. Available online: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (accessed on 12 February 2022).
45. Godoy, D. Understanding Binary Cross-Entropy. Available online: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a (accessed on 15 February 2022).
46. Davar, H. Understanding Binary Crossentropy in Its Core. Available online: https://medium.com/analytics-vidhya/binary-crossentropy-in-its-core-35bcecf27a8a (accessed on 15 February 2022).
47. Keren, G.; Sabato, S.; Schuller, B. Fast Single-Class Classification and the Principle of Logit Separation. Available online: https://arxiv.org/pdf/1705.10246.pdf (accessed on 15 February 2022).
48. Keras. Probabilistic Losses. Available online: https://keras.io/api/losses/probabilistic_losses/ (accessed on 15 February 2022).