Received 3 March 2023, accepted 7 March 2023, date of publication 10 March 2023, date of current version 21 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3255404

RESEARCH ARTICLE

Neural Networks for Aircraft Trajectory Prediction: Answering Open Questions About Their Performance

DANIEL AYALA[1], RAFAEL AYALA[2], LARA SELLÉS VIDAL[3], INMA HERNÁNDEZ[1], AND DAVID RUIZ[1]
[1]ETSII, University of Seville, 41012 Seville, Spain
[2]Molecular Cryo Electron Microscopy Unit, Okinawa Institute of Science and Technology Graduate University, Onna, Kunigami, Okinawa 904-0411, Japan
[3]Nucleic Acid Chemistry and Engineering Unit, Okinawa Institute of Science and Technology Graduate University, Onna, Kunigami, Okinawa 904-0411, Japan

Corresponding author: Daniel Ayala (dayala1@us.es)

This work was supported in part by the Spanish Ministry of Science, Innovation and Universities under Grant PID2019-105471RB-I00; and in part by the Office for Economy and Knowledge of the Andalusian Regional Government under Grant P18-RT-1060 and Grant US-1380565.

ABSTRACT The increase in air traffic in the recent years has motivated the development of technologies to monitor air space and warn about possible collisions by predicting the trajectories that will be followed by aircraft. In this field, neural networks have become prominent thanks to their potential to learn to predict maneuvers without providing aspects that are difficult to model such as atmospheric conditions, or detailed aircraft parameters. A variety of models have been proposed; however, these are often tested in very limited setups, leaving many unanswered questions about how they perform in certain conditions, or whether or not their accuracy can be improved by training models for specific trajectories, using additional features, predicting more distant points directly, etc. This may be problematic for researchers or developers of these systems, who have no way of knowing what strategies will yield the best results. We have identified ten open research questions that have not been answered through in-depth testing. This motivated us to carry out a novel experimental study that performs aircraft trajectory prediction with several dozens configuration variants to answer the aforementioned questions by means of a much more complete evaluation. Some of the conclusions of our study stand in contrast with some popular practices in the state of the art, which casts some doubts on the simplicity of their application; for example, differential features are crucial for proper performance but are not mentioned by most studies, while complex, more elaborate models may lead to worse results than simple ones. Other important insights include the benefit from specialized models in more challenging scenarios, the influence of the known trajectory length in said scenarios, the step degradation of predictions when predicting further into the future, or the detrimental effect of adding additional features. These insights should help guide future research about the application of neural networks when it comes to aircraft trajectory prediction and their eventual inclusion in final systems.

INDEX TERMS Aircraft, neural networks, trajectory prediction.

I. INTRODUCTION

In the recent years there has been a pronounced development and increased presence of the air transport industry, resulting in increased air traffic [1]. This has led to a higher necessity of systems that can assist controllers in making decisions related to the flight, e.g. detecting aircraft faults [2], monitoring the air space to warn about possible collisions [3], [4] or even apply changes in the trajectory [5]. These systems usually involve forecasting the state of aircraft in the airspace for safety purposes [6]. The knowledge of the airspace (position and movement of the aircraft in an area) is possible thanks to the adoption of technologies such as Automatic Dependent Surveillance-Broadcast (ADS-B) and Mode S, which provide

The associate editor coordinating the review of this manuscript and approving it for publication was Rosario Pecora.

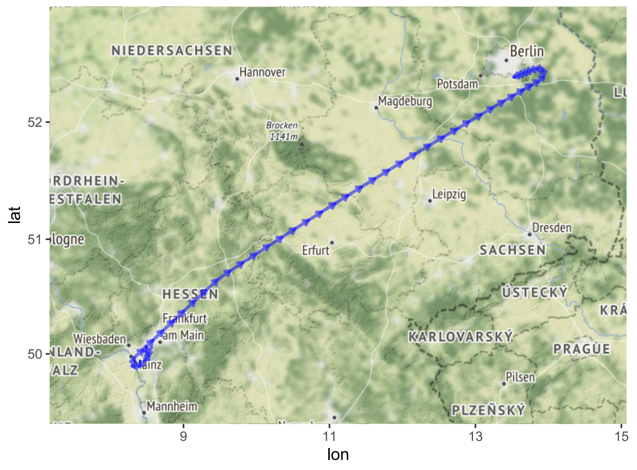


FIGURE 1. Flight example.

aircraft information over publicly accessible radio channels that can be accessed by other aircraft or ground stations. These technologies involve the periodical emission of signals hat either include precise location information about the aircraft, typically determined by a Global Navigation Satellite System, or that can be used to determine the location through multilateration.

The usefulness of trajectory prediction for monitoring systems is evidenced by the recent development of FLARM (''Flight Alarm'') messages [7], which transmit a prediction of the trajectory of the aircraft in the next 20 seconds produced by some independent algorithm. It is therefore reasonable that a significant amount of research has focused on developing techniques for the prediction of aircraft trajectories [8], that is the prediction of the three-dimensional position of a plane in future time instants. Among the existing techniques to aid in this process, a modern approach is the application of neural networks, which can potentially learn complex aircraft flight patterns without needing to be fed explicit knowledge about maneuvers, pilot intentions, or route information. While many trajectories are trivial to predict, some parts of the flight, specially during the ascent or descent of the aircraft, and at turning points, become irregular. For example, Figure 1 shows the trajectory of a flight from Frankfurt to Berlin in which the most irregular parts can be observed during takeoff and landing. Neural networks could be able to learn to a certain point the factors that govern the movement of the aircraft, which would otherwise require complex, explicit modelling: area geography, atmospheric conditions, flight phase, etc. For example, it may be typical for flights to turn towards an airport when flying over a certain geographical area.

While the aforementioned factors and expert knowledge are fundamental parts of a final prediction model, neural networks may provide additional information or corrections in challenging scenarios in which trajectory dynamics are harder to model, little information is available, or computational methods are too expensive. The promising performance of neural networks, along with the availability of large amounts of data required for their training have motivated many authors to study the application of neural networks for aircraft trajectory prediction [8]. However, such studies usually involve the evaluation under a single, simple configuration, such as using a fixed number of former points for the prediction, or only predicting the immediate next point of the trajectory. This offers little insight that may be actually useful to someone who wants to implement an aircraft trajectory prediction system in a real-world context, since questions like the following arise: does it make a significant difference to use more former points for a trajectory? How much do sequential predictions degrade as we make them based on previous ones? Are results better when using specialised models? Do more complex models actually achieve better performance?

To shed more light on these issues and properly study the performance of neural networks to aircraft trajectory prediction we have carried out a comprehensive experimental study that tests the performance of trajectory prediction under several dozens of different configurations to answer ten open research questions of significant importance when it comes to the application of neural networks in real-world scenarios: 1) Do more complex architectures really achieve a significantly better performance? 2) Do neural networks achieve better performance than simpler baselines that do not require training? 3) To what extent does the length of the input trajectory impact performance? 4) When performing multi-step trajectory prediction, how does the prediction accuracy degenerate as we make more sequential predictions from the former ones? 5) To predict a distant position, is it better to perform multi-step prediction or delayed prediction? 6) Is it better to use differential or absolute features? 7) How do models perform when trained with (and applied to) specific flight phases? 8) How do models perform when trained with (and applied to) trajectories with less regular movement (more turns)? 9) Does performance improve if additional trajectory-wide features related to the position of the plane in the flight are added? 10) Does performance improve if we use extra state vector features in addition to positional ones (latitude, longitude, and altitude)?

Our experiments have been structured in six scenarios, in each of which a comparison is performed to help answer one or several questions. Our contribution to the state of the art can therefore be summarized as follows: we present the results and conclusions of an in-depth exploratory study aimed at evaluating the capabilities of neural networks when used to predict aircraft trajectories in isolation (that is, without using information beyond the available state vectors describing the trajectory). This study should provide useful information about what strategies are more promising and under what circumstances they struggle to even broadly predict trajectories.

The next section describes concepts necessary to properly understand the rest of our work. Section III describes the related work on aircraft trajectory prediction with a focus on how neural networks have not been properly evaluated in-depth. Section IV describes the motivation and scope of our research. Section V contains our experimental study, including the open questions we have identified, the network architectures we have considered, the experimental setup, and discussion about the results. Section VI summarizes our contributions and the results of our study.

II. PRELIMINARIES

In this Section, we describe terms that are relevant to the problem of trajectory prediction.

- **Flight** A flight is the movement of an aircraft from an origin airport to a destination airport. The three main phases of a flight are takeoff, cruise, and landing, corresponding to the ascent from the origin, the movement at relatively constant height, and the descent into the destination.
- **State vector** A state vector is a collection of features describing the state of an aircraft at a given instant during a flight. Said features can include the position of the aircraft (latitude, longitude, and altitude), its identifier, the movement angle, etc. We denote the value of feature f for a state vector v as $v.f$, so for example $v.time$ refers to the instant corresponding to the state vector, and $v.lat$ to the latitude of the aircraft at said instant.
- **Trajectory** A trajectory is a sequence of n equally distant state vectors ((e.g. a state vector every 30 seconds) corresponding to the flight of an aircraft during a period of time. Said state vectors should include, at the very least, the position of the aircraft. We denote the n state vectors of a trajectory t as $t = \{v_1, v_2, \dots, v_n\} | v_i \in V$. We refer to the amount of time between state vectors as the period of the trajectory, denoted by $p(t)$. Trajectories are used as inputs for trajectory prediction. A longer trajectory means, independently of the duration of the flight, how much information is available about past points of the aircraft, be it because observations include a larger amount of time, or because the temporal resolution is higher. Figure 2 depicts how a trajectory of length $n = 4$ is used to make predictions.
- **Differential features** A differential feature is a derived featured computed as the increment of a numerical feature with respect to its value in the former state vector of the trajectory. We denote them as $v_i.\Delta f = v_i.f - v_{i-1}.f$. The first state vector of a trajectory has no differential features since a former vector is needed to compute the difference. These features provide information about the evolution of features in a more explicit way. Figure 8 shows how the three positional features (latitude, longitude, and altitude) are transformed into differential features.
- **Trajectory-wide features** A trajectory-wide feature is a feature associated to an entire trajectory rather than a single state vector. For example, information about the physical properties of the aircraft, the weather, or the position of the destination.

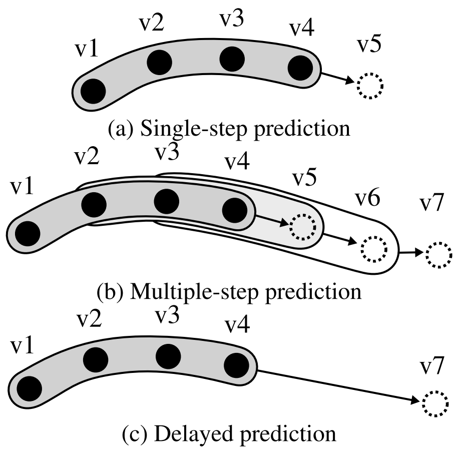


FIGURE 2. Prediction types.

- **Single-step trajectory prediction** Single-step trajectory prediction (Figure 2a) consists in predicting the next state vector of an aircraft (the one after a single period), including at the very least its position, from a given trajectory of $n > 0$ state vectors. $v_{n+1} = ssp(\{v_1, v_2, \dots, v_n\})$, where $ssp(t)$ denotes the function that performs single step prediction for trajectory t, corresponding to the state of the aircraft at time $v_{n+1}.time = v_n.time + p(t)$. The predicted state vector does not necessarily have to include the same features as the original ones, but the latitude, longitude, and altitude of the aircraft are essential.
- **Multi-step trajectory prediction** Multi-step trajectory prediction (Figure 2b) consists in repeatedly applying single-step prediction to predict several state vectors from a trajectory, allowing the prediction of the aircraft position an arbitrary amount of time into the future. Once v_{n+1} has been predicted, then $v_{n+2} = ssp(v_2, v_3, \dots, v_{n+1})$, that is, the first state vector is discarded and the new one is appended at the end of the trajectory. Note that in order for multi-step prediction to be feasible, the predicted state vectors must contain all the features included in the input in order to be usable as part of the next input.
- **Delayed trajectory prediction** Delayed trajectory prediction (Figure 2c) consists in predicting the state vector of an aircraft after several periods from a given trajectory of n state vectors. We refer to the amount of periods into the future as the offset o. For example, we would express the prediction of the 5th following state vector as $v_{n+5} = dp(\{v_1, v_2, \dots, v_n\}, 5)$, where $dp(t, o) | o > 1$ denotes the function that performs delayed prediction for trajectory t with offset $o \geq 0$, corresponding to the state of the aircraft at time $v_{n+o}.time = v_n.time + p(t) \times o$. A limitation of delayed trajectory prediction is that it makes it impossible to directly perform multi-step

predictions from the output, since the intermediary state vectors would be missing.

- **MAE** The Mean Absolute Error (MAE) is a metric of prediction quality that measures the average difference between a prediction and the ground truth. In the context of trajectory prediction, said difference is the euclidean distance between the predicted features and the actual ones, though only the features related to the position should be taken into account to convey the predicted movement accuracy. Before computing the MAE, the coordinates of the plane should be translated into a homogeneous frame such as the Earth-centered, Earth-fixed (ECEF) coordinates system, which takes the static center of the Earth as the origin of the coordinates system.

III. RELATED WORK

Methods for modelling aircraft motion to perform trajectory prediction have been classified into three categories [8]: state estimation methods, kinetic methods, and machine learning methods.

State estimation methods [9], [10], [11], [12], [13] propagate the motion of an aircraft (position, speed, etc.) according to mathematical models based on state transition and error variance matrices built from past observations. The estimations can be made with a single or multiple models. When using multiple models, several flight modes are defined, and a model is used to predict motion in each of them. The transition between modes is modelled with a Markov process with a transition probability matrix that depends on the aircraft's state. These methods do not take into account information about the aircraft such as its mass or shape. For example, Dalmau et al. [9] aim to predict the current intents of an aircraft by inferring its guidance mode (e.g. descent at constant Mach and idle thrust) through a multiple-model algorithm focused on handling vertical-movement maneuvers through the aggregation of several different mathematical models. Lin et al. [11] predict the time and altitude at which an aircraft reaches planned route segments by applying a Hidden Markov Model in which each segment of the route is a hidden state, and past observations are used to adjust the state transition probabilities. Choi et al. [13] present an hybrid method that first applies a machine learning algorithm to predict the next position of the aircraft, and then corrects said point by applying an Stochastic Linear Hybrid System that models both continuous variables (latitude, longitude, etc.) and discrete ones (flight modes).

Kinetic methods [14], [15] model the multiple forces acting on an aircraft through differential equations that handle the acceleration caused by said forces, which include gravity, engine thrust, or drag. Their effect depends on a number of factors related to the aircraft such as its mass, position, altitude, speed, or orientation, as well as external ones such as atmospheric conditions or wind strength. Apart from these variables, the pilot's intentions must be known to predict the future movements of the aircraft. As a consequence, these models are parameter-heavy, and require a considerable knowledge about both the plane, the maneuvers being performed and the environment in which the flight takes place. This has prompted the development of databases to store relevant information such as the Base of Aircraft Data [16], as well as formal languages to express aircraft intentions [17]. Some of these parameters, however, can be adjusted by mining past flights data [18]. For example, Zhang et al. [14] present a model that relies on online connectivity, involving variables such as the precise angle of the plane, wing surface area, aircraft intent (as instructions split into four groups: speed, altitude, thrust or lateral), pressure, wind speed, temperature, etc.

Machine learning methods leverage large amounts of historical data to learn parameters that minimise the error when making predictions. In theory, given enough examples and relevant features, these methods can learn the underlying laws, routes and maneuvers that govern aircraft movement rather than having to manually integrate them. While relatively simple models such as regression ones can be used to predict trajectories [19], [20], the models that have caught the most interest are those based on neural networks. The flexibility and capability of neural networks to approximate arbitrary functions and exploiting low-level features make them ideal for a context where the output may be influenced by complex, conditional non-linear combinations of the input. Furthermore, neural network architectures like LSTM, GRU or convolutional layers are suitable to deal with the typical sequential nature of the data (the consecutive measurements of the aircraft's state), which make them a popular choice in the field. Since neural networks are the focus of our work, next we describe individual proposals. However, rather than being interested in the model themselves, our interest lies in studying what research questions are answered in their experimentation, and therefore we enumerate the testing conditions of each work.

Similar methods can be used for the prediction of the trajectory of other vehicles such as ground vehicles [21], [22], [23], which is useful for predicting traffic conditions, collision risks, route optimization etc. The common elements (prediction of the location of a vehicle) lead to the use of similar models such as Hidden Markovv Models [21], neural networks that leverage positional features [22], [23]. However, the differences in the application context make those models unapplicable to aircrafts. For example, the movement of vehicles is 2-dimensional and usually restricted to roads, which lead to a stronger focus on network-related algorithms and path-finding.

A. TESTING CONDITIONS OF NEURAL NETWORKS IN THE STATE OF THE ART

Wu et al. [24] test the single-step application of a densely connected neural network to predict the next state vector from the last 3 observations. Min et al. [25] test the single-step application of a recurrent Elman network to predict the next two state vectors from the last 4 observations. Shi et al. [26] test the single-step application of a LSTM network to predict the next state vector from the last 10 observations. Xu et al. [27] test the multi-step application of a LSTM network to predict the next 30 state vectors from the last 20 observations. This work focuses on the prediction of the trajectories of several aircraft simultaneously. Zeng et al. [28] test the single-step application of a LSTM network to takeoff and landing trajectories. The network predicts the next 90 and 150 seconds of the trajectory for takeoff and landing samples respectively (the number of points that correspond to this time period is not mentioned) from the last 35 and 48 observations respectively (reported to be the optimal value when testing in the 10 to 60 range). Yang et al. [29] test the multi-step application of a LSTM network to predict the next five state vectors from the last 3 observations. Shi et al. [30] test the single-step application of a LSTM network to predict the next state vector from the last 10 observations, splitting the dataset into climbing, cruising, and descending phases. Ma et al. [31] test the single-step application of a LSTM network to predict the next state vector using differential features. Ma and Tian [32] test the multi-step application of a network that combines the application of 1-dimensional convolutions to the state vectors sequence and LSTM layers. The network is used to predict the next 5 state vectors from the last 6 observations, reporting the error for the predictions at positions 1, 3, and 5.

Table 1 contains a summary of the testing conditions of the former proposals. Our conclusions from the study of the state of the art is that while many (often similar) models have been proposed, there is little consideration about researching how certain factors or variants may affect prediction. Only three of the aforementioned proposals [27], [29], [32] test the application of multi-step prediction, none of them report the effect of using different trajectory lengths for prediction, and only one of them [31] considered the use of differential features. In a real use-case, these works offer few insights on what strategies and practices can offer the best results, and what information can benefit the models.

IV. MOTIVATION AND SCOPE

Our research is limited to the evaluation of neural network models when applied to the prediction of aircraft trajectories in an unknown context, that is, without any information beyond the trajectories themselves. Our purpose is to compare the effect of different strategies on performance to evaluate their predictive potential rather than present the use of neural networks as a final solution for performing trajectory prediction.

While it may be tempting to see neural networks as a generic solution to all prediction-related problems that replaces the need to model a complex system with merely feeding large amounts of data for training, neural networks are often not precise enough to replace carefully built specialized models in real, highly demanding use-cases. However, neural networks have potential to model complex unknown factors through training that are hard if not impossible to model otherwise (e.g. complex object detection in images or speech synthesis), and could help improve such models when used as a part of them or in contexts with restrictions that make their full use impossible. Doing so requires awareness of what conditions or variants lead to better results and what are the limitations of their application in this context.

The existing literature may lead to an overly optimistic view of the capabilities of neural networks given the reported results, but as we showed in Section III, existing research does not delve into an in-depth study of different considerations and therefore does not offer the insight that would be needed to eventually integrate neural networks as part of a full-fledged system.

It is precisely that lack of deeper insights that we try to partially solve with our research, whose results should not be seen as a proposal for final prediction, but as an exploration of the effect of different circumstances and strategies on the performance of neural networks in order to identify the most promising lines of future development and the cases in which they are not usable due to poor performance.

With regards to the questions themselves, we do not intend to fully cover all possible experiments and variants, since each question would require an additional in-depth study focused on it to properly do so. The scope of our work is limited to a set of experiments whose results allow us to give a general answer while identifying possible future work.

V. EXPERIMENTAL STUDY

In this section we describe the design of the study we have carried out, as well as the results we obtained. In Section V-A we define the goals of the study, namely the open research questions we have identified. In Section V-B we describe the models we use. In Section V-C we describe how we obtained the data used in the study. Finally, in Section V-D we show the experimental results we obtained in several scenarios aimed to answer the research questions.

A. GOALS

Following our study of the State of the Art, we have identified several questions that have been left completely or partially unanswered with the limited experiments performed as part of the evaluation of existing proposals. Every question represents an aspect where further experiments can be carried on to determine the effect of certain conditions or parameters:

Q1 **Do more complex architectures really achieve a significantly better performance?** Is it better to use a simple network or a more complex one? More complex architectures are those with more layer types, different parameter types, and structures designed to deal with more specific kinds of data, e.g. an architecture that uses convolutional and LSTM layers to deal with specific kinds of sequential data is more complex than a generic

TABLE 1. Summary of testing conditions of neural networks in the state of the art.

Proposal	Architecture	Input length	Prediction	Observations
Wu et al. [24]	Dense network	3	Next SV	-
Min et al. [25]	Recurrent Elman network	4	Next 2 SVs	-
Shi et al. [26]	LSTM network	10	Next SV	-
Xu et al [27]	LSTM network	20	Next 30 SVs	-
Zeng et al. [28]	LSTM network	35/48	Next 90/150s	Takeoff/landing trajectories
Yang et al. [29]	LSTM network	3	Next 5 SVs	-
Shi et al. [30]	LSTM network	10	Next SV	Climbing/cruising/descending trajectories
Ma et al. [31]	LSTM network	-	Next SV	Differential features
Ma et al. [32]	Convolutional LSTM network	6	Next 5 SVs	-

dense network. While more complex architectures may have more potential to produce complex, non-linear outputs, the added complexity can reach a point that is detrimental for the training of the network. In some contexts, simple neural networks have obtained better results than specialized architectures [33].

Q2 **Do neural networks achieve better performance than simpler baselines that do not require training?** Are there cases where the differences are more significant (that is, the neural network model is more suitable)? While a neural network is able to learn how to predict movement during maneuvers, if the movement of an aircraft is relatively regular, a baseline that extrapolates movement in a simple way may be more reliable on average than a neural network that is more susceptible to producing results with high variability.

Q3 **To what extent does the length of the input trajectory (that is, the number of state vectors used to predict the next one) impact performance?** Are longer input trajectories more beneficial under some circumstances? In an intuitive manner, providing more information about the aircraft past states' should be beneficial, but these benefits could be larger in certain cases. For example, predicting a trajectory could require the identification a flight pattern whose detection requires a minimum number of past observations.

Q4 **When performing multi-step trajectory prediction, how does the prediction accuracy degenerate as we make more sequential predictions from the former ones?** Each prediction introduces an error that affects future predictions, potentially reaching a point where the model is only fed highly erroneous observations, leading to useless predictions. If this is the case, only the first predictions of a model would be trust-worthy and different techniques should be used for long-term predictions.

Q5 **To predict a distant position, is it better to perform multi-step prediction or delayed prediction?** One way to mitigate the accumulation of errors to predict the distant future may be to directly predict the position at the desired future instant rather than reach it in several accumulated steps. Models that make such predictions are less flexible: they can't be re-applied by concatenating the output to the input state vectors, since the time difference between the last observation and the input would be much greater than the difference between the input observations. Therefore, it remains to be seen if training specific models leads to significantly better performance that justifies the loss of flexibility.

Q6 **Is it better to use differential or absolute features?** Differential features greatly limit the range of input and predicted features, making it easier for the model to learn how the plane will mode rather than at what absolute position it will be, the later being much more difficult to predict in a precise way, since an error of, for example, 1% with respect to the expected absolute position, can translate into thousands of kilometers. On the other hand, an error of 1% with respect to the expected movement would be much smaller. The advantage offered by absolute features is the additional information about the absolute position of the plane on Earth, which could correlate with certain routes, thus enabling the network to learn that at certain places of the world, aircraft tend to move in a certain way.

Q7 **How do models perform when trained with (and applied to) specific flight phases?** Since the behaviour of aircraft during different flight phases is fundamentally different, training a generic model that works well in all of them could be unrealistic. Training specialized models could help improve performance at the cost of relying on the correct identification of the current flight phase. While there may exist different proposals on how to split a flight into phases or what defines such phases, it is common to at least consider three of them: take-off, cruise, and landing.

Q8 **How do models perform when trained with (and applied to) trajectories with less regular movement (more turns)?** Independently of the flight phase, trajectories with more irregular movement (e.g. sudden changes in direction) should be more challenging, specially if most of the training data consists of samples where the aircraft follows a straight line. Similarly to Q7, specialized models could improve performance at the cost of lessening the ease of application.

Q9 **Does performance improve if additional trajectory-wide features related to the position of the plane in the flight are added?** Such features would provide information that is not associated to each state-vector, but to the entire input trajectory, such as the current phase of the flight, the position of the destination, or the current absolute location of the aircraft on Earth. The latter, for example, would help provide the aforementioned advantage of absolute features without relying on them.

Q10 **Does performance improve if we use extra state vector features in addition to positional ones (latitude, longitude, and altitude)?** While the three positional ones are always readily available, ADS-B messages may include additional fields in their state vectors, such as the flight angle and speed. These features can be used to expand the state vectors fed to the network, providing extra, explicit information that may improve performance. However, to keep the possibility of multi-step prediction, these features must also be predicted, which may be too demanding and lead to an impoverishment of the prediction of the positional features.

Section III-A shows that no study so far has attempted to systematically evaluate the performance of neural networks under a variety of conditions that can be used as reference for future research.

B. MODELS

To answer the aforementioned questions, we need to test several fundamentally different models under different circumstances to study the differences in performance. While endless models can be defined, our intentions are to test a small amount of models that represent significantly different approaches and levels of complexity. We settled for three neural network models and two baselines. The neural network models are the following: a densely connected network (DCN) as a simple architecture, a LSTM network (LSTM) as an architecture with specific mechanisms to deal with sequential data, and a CNN-LSTM network (CNN-LSTM) as a more complex and sophisticated architecture. The baselines include a spline extrapolation baseline (SPLINE) and an average extrapolation baseline (AVG).

Each model must take as input data a trajectory of length n and predict at the very least the positional features at v_{n+1}. Therefore, if the number of features of each state vector is nf, the shape of the input will be (n, nf) (two-dimensional), and the shape of the output (nf) (one-dimensional). The neural network models are also able to predict v_{n+o} where o is the offset of a delayed prediction. Next, we describe each model in detail.

1) DCN

A densely connected network uses mainly densely connected layers, which connect a fixed number of inputs features with a fixed number of output features using all possible connections (hence the name) with a parametric weight for each of them. While simple, the large number of connections in these networks makes them potentially able to learn complex transformations and combinations of data when placed sequentially. However, the high number of parameters makes the layers harder to train. Dense networks are usually applied to problems in which the nature of the features does not include concepts like sequentiality or regionality, such as semantic typing [34] or person classification [35]. Overall, they are a conceptually simple, general-purpose architecture whose main disadvantage is the high number of parameters. Despite their simplicity, densely connected networks have obtained better results than other, more sophisticated architectures in some contexts [33].

Since these layers take as input one-dimensional data but the network must process a sequence, in order to be fed sequential data of shape (n, nf), the input has to be flattened into a vector of shape $(n \times nf)$. Therefore, the input size of the first layer is determined during the creation of the network from the number of features and the trajectory size.

Our implementation of a densely connected network does this in the first dense layer, whose output is then fed to further dense layers. The final dense layer has an output size equal to nf, corresponding to the predicted features of the future state vector. The dense layers with the exception of the last one (since it produces the final prediction) are followed by the application of the tanh function to enable non-linearity, which was chosen to avoid significant information loss by preserving the negative values that are common in the input and output data, unlike other activation functions such as ReLU, which removes them.

Figure 3 depicts the final architecture of our dense network including the dimensions of the data as it goes the network using an example input with $n = 4$ and three features, highlighting the input and output data. We used five dense layers with progressively reduced output sizes.

2) LSTM

A LSTM network uses as its main component one or several Long Short Term Memory layers, which are designed to consume sequences of data and produce an output for each element in the sequence. Each element from the sequence is combined with an internal state of the layer (which is also updated with every application) and the output of the former element, all through operations that involve trainable parameters. Therefore, the output corresponding to the last element of the sequence, as well as the inner state after the last application, will be influenced by the entire sequence, making LSTM layers ideal to process sequential data. Since a LSTM layer produces an output for each input element, an input of shape (n, nf) will result in an output of shape (n, x) where

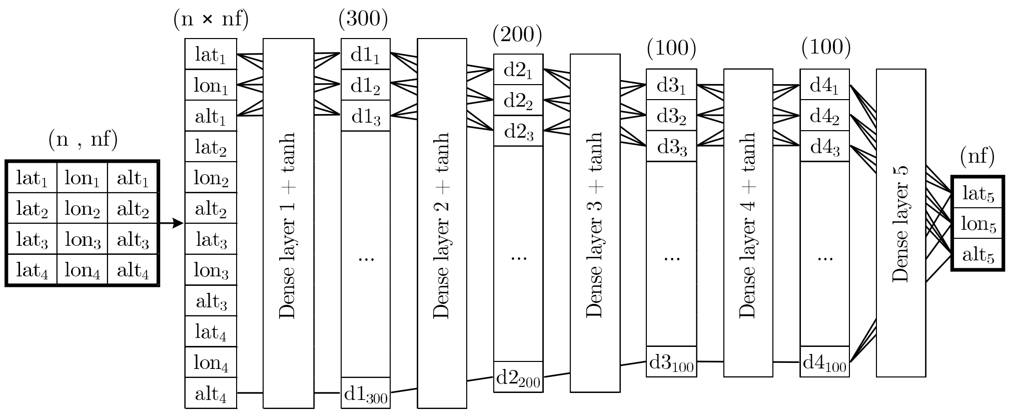


FIGURE 3. **Dense network architecture.**

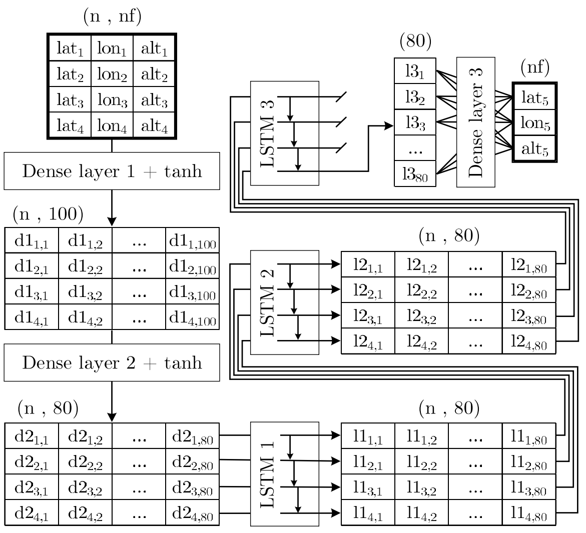


FIGURE 4. **LSTM network architecture.**

x is the configurable size of the output features, enabling the concatenation of several layers. When the last layer is applied, in order to obtain a single vector that represents the entire sequence, it is typical to retain the output or inner state corresponding to the last element of the sequence and ignore the rest.

LSTM networks have been applied to all kind of problems where the input contains some kind of sequence, such as sequences of words from a text that must be classified [36], audio signals [37], DNA sequences [38] or image sequences from a video [39]. The only requirement is that each element of the sequence must be represented numerically.

In addition of the LSTM block, our implementation of a LSTM network previously applies a block of dense layers to learn a first transformation of the input features. The LSTM block takes its outputs and returns the output for the last element of the sequence, which is then used to make the prediction through a final dense layer.

Figure 4 depicts the final architecture of our LSTM-network. We used an initial block of two dense layers, a LSTM block with three concatenated layers, and a final dense layer, highlighting the input and output data. Note that the input of the first block of dense layers has not been flattened as happens in the densely connected network; this is because the dense layer is applied independently to each row in a batch-like fashion, so that, in the first dense layer, $d1_{i,j} = D(\{lat_i, lon_i, alt_i\})|i \in [1, n], j \in [1, 100]$, where D denotes the application of the dense layer (and tanh function).

3) CNN-LSTM

A CNN-LSTM network combines the previously described LSTM layers with convolutional layers. A convolutional layer can be applied to features that have some kind of regionality which makes it possible to tell when two features are "nearby". Their purpose is to transform the input features into new ones where each original feature has integrated the nearby ones, representing information related to the "area" of features around the original one, e.g. whether or not there is a sudden change in some feature. This is usually done with a sliding window (called kernel) of a given size that traverses one or several dimensions of the input data and aggregates the values across that dimension by adding them up after multiplying them by the weights associated to each position of the kernel, which are trainable parameters.

These architectures are typical in contexts that deal with sequences of continuous features that may benefit from the pre-processing provided from the convolutional network. For example, a 1-dimensional convolution could be applied to an audio signal [40] or stock prices [41] so that the resulting features may contain explicit information about certain signal behaviour such as sudden drops, or smoothed averages through kernels that cover small regions of the sequence. These features would then be passed to the LSTM layers that produces an output that accounts for the entire sequence. Convolutions of 2-dimensional data are usually applied to contexts where the elements of the sequence can

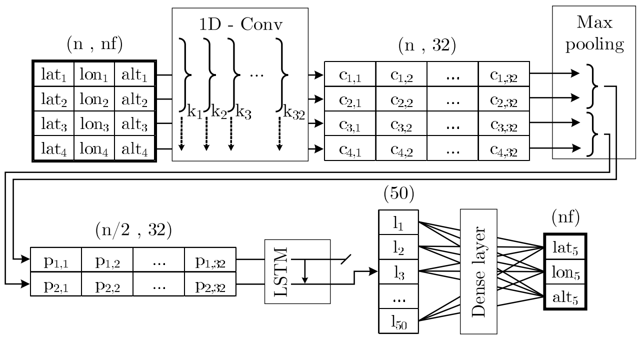


FIGURE 5. CNN-LSTM network architecture.

be represented as an image, such as traffic prediction [42] or video [43].

In the context of trajectory prediction it is possible to apply a 1-dimensional convolution to the dimension of time, that is, aggregating each feature with its value at the former and following state vectors in the trajectory.

In a CNN-LSTM network, convolutional features are first extracted, and then are passed to a LSTM layer. We implemented the architecture described by Ma and Tian [32], which first applies two convolutional layers with 32 kernels each, and then two LSTM layers, ending with a dense layer. After the convolutional layers, max-pooling is applied, which applies a sliding window that retrieves the maximum value of the covered features.

Figure 5 depicts a simplified version of the architecture with a single convolutional and LSTM layers, where k_i denotes kernel i, which traverses the time dimension and produces the i^{th} channel of the output.

4) SPLINE

Splines are functions defined from (x, y) observations that go over such observations in a smooth way. Splines are usually used to interpolate y for a value of x between two of the observations. However, splines can also be used to extrapolate the value of y beyond the last value of x. To apply splines for trajectory prediction, we create a spline for each state vector feature, using the position in the trajectory as the x variable, and the value of the feature as y. Then, we extrapolate the value of each spline for the next value of x.

5) AVG

The averaging baseline only works with differential features, as it predicts that the movement of the aircraft will be a weighted average of the last movements. Concretely, it computes $sv_{n+1} = 0.5\ sv_n + 0.3\ sv_{n-1} + 0.2\ sv_{n-2}$.

C. DATA ACQUISITION AND PRE-PROCESSING

To obtain data about flight trajectories we used the R package openSkies [44], which offers easy programmatic access to the OpenSky network resources [7], which provide free access to high-quality air traffic data obtained through a network of

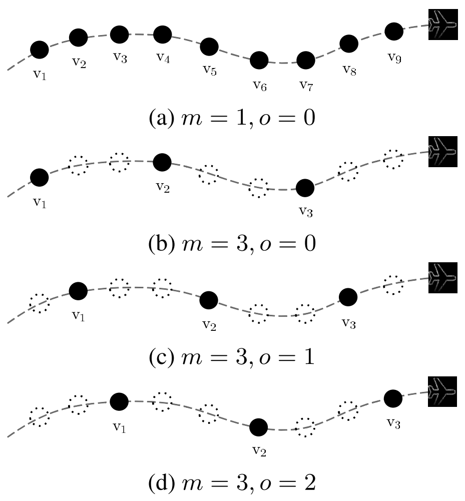


FIGURE 6. Application of different subsampling parameters to the same trajectory.

thousands of ground stations that receive ADS-B messages broadcast from the aircraft themselves for their tracking.

The openSkies package allows performing queries that retrieve flights with the state vectors that represent their entire flight trajectories with a given period. To obtain an initial set of flights, we queried for the flights that departed from the Frankfurt airport (EDDF) during the first of January of 2021 with a period of 30 seconds, and kept those that had attached state vectors, resulting in 861 flights with a total of 247070 state vectors. Each state vector contains a variety of features. However, several of them are not useful, such as a flag indicating whether or not the aircraft is on ground or the squawk code used for communications, or are derived from other features, such as the aircraft speed. For these reasons, we use as base features the latitude, longitude, and barometric altitude, which also have the highest availability. Next, we describe the process for obtaining the final dataset from the collection of flights, which is also shown in Algorithm 1.

Step 1: We subsample the state vectors of every flight using a period multiplier that determines the period of the consecutive state vectors. A multiplier of 1 means all state vectors will be used, resulting in a period of 30 seconds. A multiplier of 3 means one of every three vectors will be used, resulting in a period of 90 seconds. Multipliers with value m also enable the use of an offset in the range $[0, m)$ that determines what state vectors will be used. Figure 6 shows how a trajectory would be sampled using different multiplier and offset values. We subsample using all multiplier values in the range $[1, 4]$, ensuring that the models will learn how to predict from arbitrarily spaced points, avoiding overfitting to a specific period. For each multiplier value, we create samples using all possible offset values.

Step 2: We take trajectory fragments as final evaluation samples using a sliding window with a size determined by the number of state vectors that will be used for the predictions, n, and a fixed number of future state vectors that determine

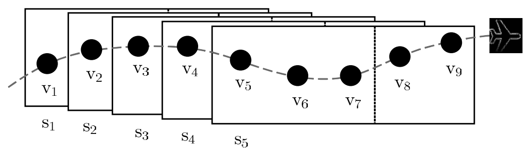
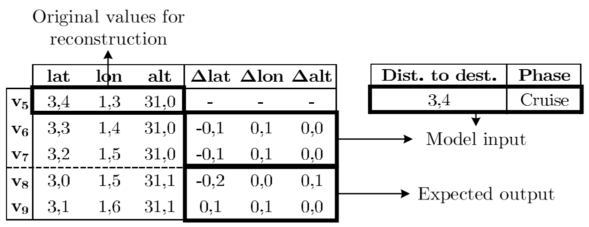


FIGURE 7. Samples example with $n = 3, f = 2$.



	lat	lon	alt	Δlat	Δlon	Δalt
v_5	3,4	1,3	31,0	-	-	-
v_6	3,3	1,4	31,0	-0,1	0,1	0,0
v_7	3,2	1,5	31,0	-0,1	0,1	0,0
v_8	3,0	1,5	31,1	-0,2	0,0	0,1
v_9	3,1	1,6	31,1	0,1	0,1	0,0

Dist. to dest.	Phase
3,4	Cruise

FIGURE 8. Features selection example.

the maximum number of future predictions, f. We remove samples in which any of the state vectors is null. Figure 7 depicts an example in which 5 samples are taken from the trajectory. We take all possible fragments with $n \in [5, 10]$ and $f = 20$. While in our experiments we only predict up to the 5th future state vector, we store a larger number to include more information about the trajectory.

Step 3: We select and retain the features that will be used for each sample. If at least one feature is differential, the first state vector will be stored to enable the reconstruction of the original value of the features and the sequence size will be reduced by 1 (since the first state vector has no increment). Apart from the resulting sequence of state-vectors, trajectory-wide features can be included, such as the distance to the destination at the first state vector of the phase of the flight. Figure 8 depicts an example in which the features selected were the differential latitude, longitude and altitude, as well as the distance to the destination and the flight phase as trajectory-wide features.

Step 4: We remove anomalous samples, which are common given the noisy nature of the data from ADS-B messages. We remove samples where any of the features had a null value, where any of the sequence steps had no increment for all features (probably caused by the repetition of a state vector), or where any of the feature values deviated more than 2 standard deviations from the mean value of the feature.

Step 5: Once all samples have been collected, we normalise the feature values (including extra features) by subtracting the mean of each feature and dividing by the standard deviation, ensuring that the mean of the normalised features will have a mean of 0 and a standard deviation of 1.

Step 6: All samples are randomly sorted using a fixed seed for reproducibility and then split into training and testing sets. The first half of samples is used for training, and the second one for testing, which is reasonable given the large amount of samples (several millions after pre-processing). The random ordering of the samples should remove the possibility of biased training where results are affected by the presence or omission at training or testing of samples from flights with particular characteristics.

Algorithm 1 Data Preprocessing

```
Input: $F$: Set< Flight>   Collection of flights $M$:
       Set< Integer>   Subsampling multipliers $N$:
       Set< Integer>   Trajectory samples lengths $f$:
       Maximum number of future predictions $r$:
       Integer   seed for randomization of samples
       order
Output: $Tr$: List< Sample>   Collection of training
        samples $Te$: List< Sample>   Collection of
        testing samples
/* For each flight in the evaluation set */
foreach $flight$ in $F$ do
    /* For each subsampling multiplier */
    foreach $m$ in $M$ do
        /* For each sampling offset */
        foreach $o$ in $[0, m)$ do
            /* The flight is subsampled
               according to the parameters */
            $ssFlight \leftarrow subsample(flight, m, o)$
            /* For each desired trajectory
               length */
            foreach $n$ in $N$ do
                /* For each possible starting
                   point of the trajectory */
                foreach $i$ in $[0, length(sf) - n]$ do
                    /* We store the state vectors
                       corresponding to the input
                       and expected output,
                       as long as there are no
                       null or repeated state
                       vectors */
                    $fragment \leftarrow ssFlight[i : i + n + f]$
                    $sample \leftarrow getFeatures(fragment)$
                    if $noNulls(sample)$ and
                     $noRepeated(sample)$ then
                        $S \leftarrow S \cup sample$
                    end
                end
            end
        end
    end
end
/* We remove anomalous data and normalise the
   rest */
$S \leftarrow removeAnomalous(S)$
$S \leftarrow normalised(S)$
/* We create the training and testing splits
   */
$S \leftarrow randOrder(S, r)$
$Tr \leftarrow S[0 : length(S)/2]$
$Te \leftarrow S[length(S)/2 : length(S)]$
```

All the data and scripts used for pre-processing, as well as the scripts that apply the models and compute metrics have been made available online.[1]

D. EXPERIMENTAL RESULTS

In this section we present the results from different experimental scenarios designed to answer the research questions presented in Section V-A. Each scenario presents a comparison of the results obtained from different techniques that provides insights into at least one of the research questions. Table 2 shows what questions are covered by each scenario, with each question being associated to at least one scenario.

Unless specified otherwise, the state vector features used in all scenarios are the latitude and longitude in degrees, and the altitude in meters. In each scenario we show plots with all the results, and a summary table with the results when using 10 trajectory points for the first, third, and fifth predictions. This table is necessary to properly compare the results for the first predictions, in which the error is lower and harder to appreciate in the plots.

All reported results show the MAE of predictions (lower is better) after translating into the ECEF coordinate system. We use a starting learning rate of 10^{-3} that is dynamically reduced if the loss stagnates, and a batch size of 256.

1) SCENARIO 1: NETWORK ARCHITECTURE AND FEATURE TYPE SELECTION

Our first experimental scenario seeks to determine what network architecture obtains the best results, as well as whether it is convenient to use normal features or differential features. We trained a CNN-LSTM network, a DCN, and a LSTM network, using 50 epochs and testing the use of normal or differential features for all of them. Results are shown in Figure 9 and Table 3 with different y-axes scales due to the disparity when using differential and absolute features. It can be observed that using differential features results in a performance that is several orders of magnitude better than normal features, which was expected since differential features keep the value of features (both used for prediction and being predicted) more constrained and not dependant on the original and destination of the flight. Among the three networks, the LSTM model yields the best results, with the DCN probably being too simple and the CNN-LSTM model too complex. Given the results, for the rest of the experiments we will use a differential LSTM model as the base neural network model.

Regarding the degradation of the predictions when making further predictions, it can be observed that, as expected, the MAE ramps up as further predictions are made, reaching a point where predictions are just useful to have a general idea of the direction of the plane rather than its actual position. This phenomenon can be seen across all scenarios.

[1] https://github.com/DEAL-US/aircraft-trajectory-prediction

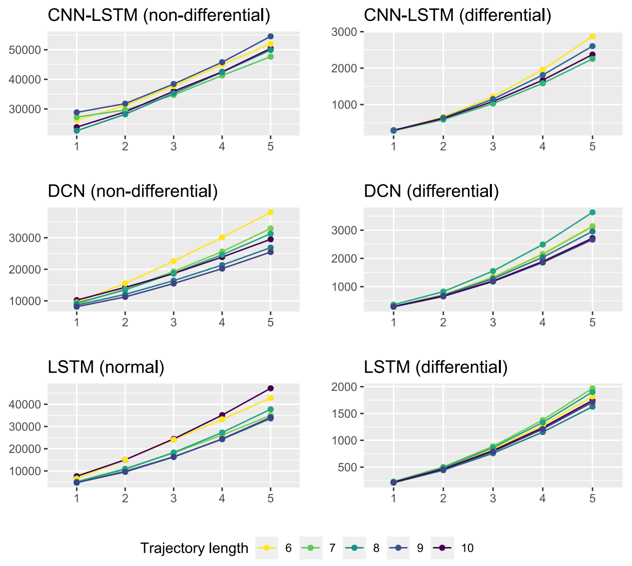


FIGURE 9. Results for S1. X-axis = prediction offset. Y-axis = MAE.

As an additional experiment, we tested the LSTM model when training with 50, 200, and 300 epochs to determine to which point it is convenient to train the network. Results are shown in Table 4. There is a significant improvement when increasing the number of epochs from 50 to 200 but not from 200 to 300. This shows that the model is neither over-fitted nor under-trained. If it was over-fitted, a lower number of epochs would have resulted in significantly better results. If it was under-fitted, the improvement would have taken place when setting a higher number of epochs. Therefore, for the rest of the experimental scenarios, neural networks will be trained with 200 epochs.

2) SCENARIO 2: BASELINE RESULTS

This scenario aims to compare the results obtained by a neural network to those obtained by simple, unsupervised baselines: the spline and averaging baseline described in Section V-B. Results are shown in Figure 10 and Table 5. The neural network obtains significantly better results than both baselines. The averaging baseline performs better than the spline one, probably because spline extrapolation can become erratic. The relatively good results of the averaging baseline are probably caused by the fact that during most of the flight the plane travels in a straight line. In some of these cases, the LSTM model tends to erroneously predict a curve as can be seen in Figure 11 in which the baseline model correctly predicts the straight trajectory, which led us to the hypothesis that using different models for different kinds of trajectories could help diminish said errors.

3) SCENARIO 3: TURNING TRAJECTORIES

This scenario tests the behavior of the neural network when trained with and applied to only the top 10% of the trajectories with the most turns. We define the “turning amount” of

TABLE 2. Research question coverage with experimental scenarios.

	S1	S2	S3	S4	S5	S6
Q1: network architecture performance	✓					
Q2: baseline performance		✓	✓	✓		
Q3: Trajectory length influence	✓	✓	✓	✓	✓	✓
Q4: multi-step prediction degeneration	✓	✓	✓	✓	✓	✓
Q5: delayed prediction performance					✓	
Q6: differential features performance	✓					
Q7: flight phase influence				✓		
Q8: trajectory irregularity influence			✓		✓	
Q9: trajectory-wide features performance						✓
Q10: extra features performance						✓

TABLE 3. Results summary for S1.

Model	Error 1	Error 3	Error 5
CNN-LSTM (non-differential)	23860.84	35968.30	50440.26
CNN-LSTM (differential)	295.43	1084.57	2372.05
DCN (non-differential)	10231.54	18683.35	29510.88
DCN (differential)	293.54	1200.43	2717.48
LSTM (non-differential)	7629.14	24421.35	47184.59
LSTM (differential)	217.17	808.58	1744.67

TABLE 4. Results summary for different numbers of epochs.

Model	Error 1	Error 3	Error 5
LSTM (50 epochs)	217.17	808.58	1744.67
LSTM (200 epochs)	159.33	534.35	1140.61
LSTM (300 epochs)	151.16	524.75	1146.50

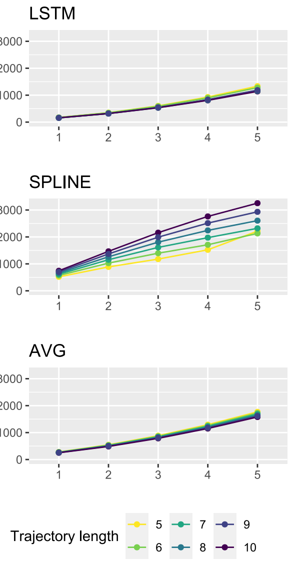


FIGURE 10. Results for S2. X-axis = prediction offset. Y-axis = MAE.

TABLE 5. Results summary for S2.

Model	Error 1	Error 3	Error 5
LSTM	159.33	534.35	1140.61
AVG	250.81	785.92	1578.25
SPLINE	743.73	2158.05	3254.05

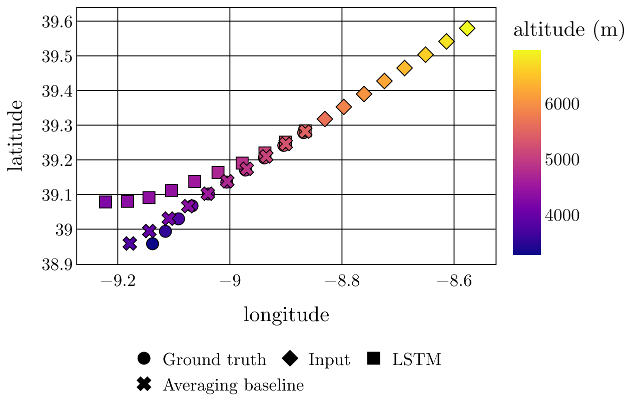


FIGURE 11. Example of LSTM and averaging baseline prediction. Interactive 3D visualization.

a trajectory as the accumulated sum of the turning degree (regarding latitude and longitude) of each point in the trajectory. Our purpose is to check whether or not we can obtain better performance by training a model that focuses on cases with more irregular movement. Results are shown in Figure 12 and Table 6, in which the LSTM (all training, turn test) model was trained with all trajectory but tested with only turning trajectories, while the LSTM (turns only) model was trained and tested with only turning trajectories. As expected the results of the LSTM model when trained with all trajectories and tested with turns only are significantly worse than those obtained from the average of all trajectories, and even worse than those obtained by the baseline. However, when a model is specifically trained with only the turning trajectories, the results obtained are much better, surpassing the baseline.

It can also be observed that the input trajectory length has a greater effect on the turning trajectories model, which is

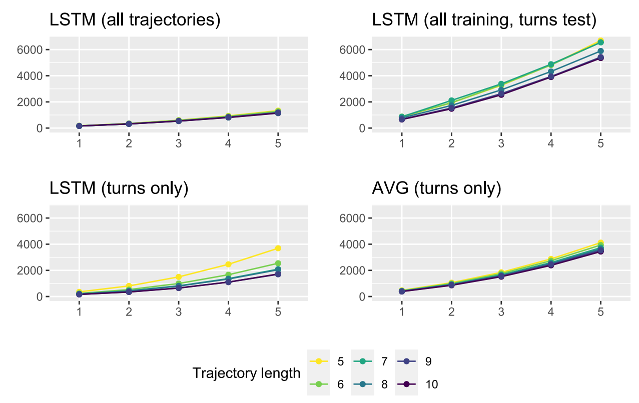


FIGURE 12. Results for S3. X-axis = prediction offset. Y-axis = MAE.

TABLE 6. Results summary for S3.

Model	Error 1	Error 3	Error 5
LSTM (all trajectories)	159.33	534.35	1140.61
LSTM (turns only)	163.47	646.93	1723.07
LSTM (all training, turns test)	655.60	2542.72	5352.20
AVG (turns only)	382.18	1517.85	3438.36

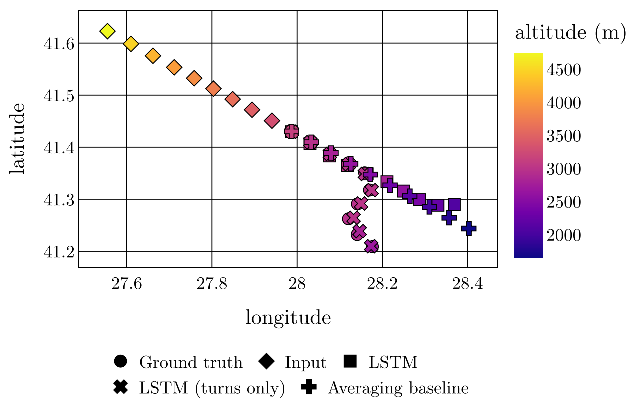


FIGURE 13. Example of predictions for turning trajectory. Interactive 3D visualization.

reasonable since the prediction of how a plane turns should benefit more from the higher amount of information.

Figure 13 shows an example in which the model trained with turning trajectories is able to predict more accurately the turn of a plane, even when the plane has not started turning yet. A possible explanation is that the trajectories with more turns correspond to key points in the trajectory of planes that tend to take place at some geographical point. The specific model is then able to learn where some of these turns take place, leading to part of the improvement.

4) SCENARIO 4: FLIGHT PHASES

This scenario tests the behavior of the neural network when trained with and applied to only trajectories in certain flight phases. Identification of flight phases is performed using a fuzzy logic approach as described by [45]. We consider four phases: climb, cruise, descent, and level (an intermediary step before descent). Our purpose is, similarly to turning trajectories, to check whether or not we can obtain better performance by training models that focus on specific flight phases. Results are shown in Figure 14 and Table 7. As expected results are usually better during the cruise phase, when the plane frequently follows a more predictable straight path and the baseline is able to obtain results on par with the other models. While the LSTM model trained with all trajectories is still better than the baseline, the models trained for specific phases outperform them in all cases, proving that it is convenient to be able to identify flight phases and use models that are specialized in that phase in particular.

Consistently with scenario 3, results in phases with reasonably higher motion variability (mainly the Climb phase) clearly improve as more trajectory points are provided.

TABLE 7. Results summary for S4.

Model	Error 1	Error 3	Error 5
LSTM (all trajectories)	159.33	534.35	1140.61
LSTM (Climb)	77.98	233.51	509.88
LSTM (Cruise)	111.35	366.64	783.84
LSTM (Descent)	101.58	463.67	1245.22
LSTM (Level)	125.36	607.76	1661.37
LSTM (all training, Climb)	176.82	578.93	1178.15
LSTM (all training, Cruise)	143.82	464.80	976.01
LSTM (all training, Descent)	198.46	852.98	2044.31
LSTM (all training, Level)	160.78	685.94	1659.73
AVG (Climb)	334.62	1211.91	2454.32
AVG (Cruise)	201.04	519.92	973.24
AVG (Descent)	440.77	1782.83	3921.97
AVG (Level)	360.14	1375.32	2861.24

5) SCENARIO 5: DELAYED PREDICTION

This scenario aims to test the effect of predicting a future position with a delayed prediction, compared to multi-step prediction. In particular, we test the prediction of the 5th future point in the trajectory. Additionally, we test for all trajectories and for turning trajectories similarly to scenario 3, since such cases could benefit more from giving a direct prediction. Results are shown in Figure 15 and Table 8. Interestingly, delayed predictions lead to worse results in the overall results, but to a significant improvement for the turns only model. In other words, delayed prediction seems to greatly benefit from separate models.

Figure 16 shows an example in which the prediction of the 5th future point is better when using a delayed prediction model.

It must be kept in mind that despite these conditional improvement, delayed prediction models lack flexibility, since they can't predict several points into the future, which makes them suitable only for some conditions.

6) SCENARIO 6: ADDITIONAL FEATURES

This scenario tests the effect of using additional features in the model beyond the basic ones (latitude, longitude, and altitude). For that purpose, we test two strategies:

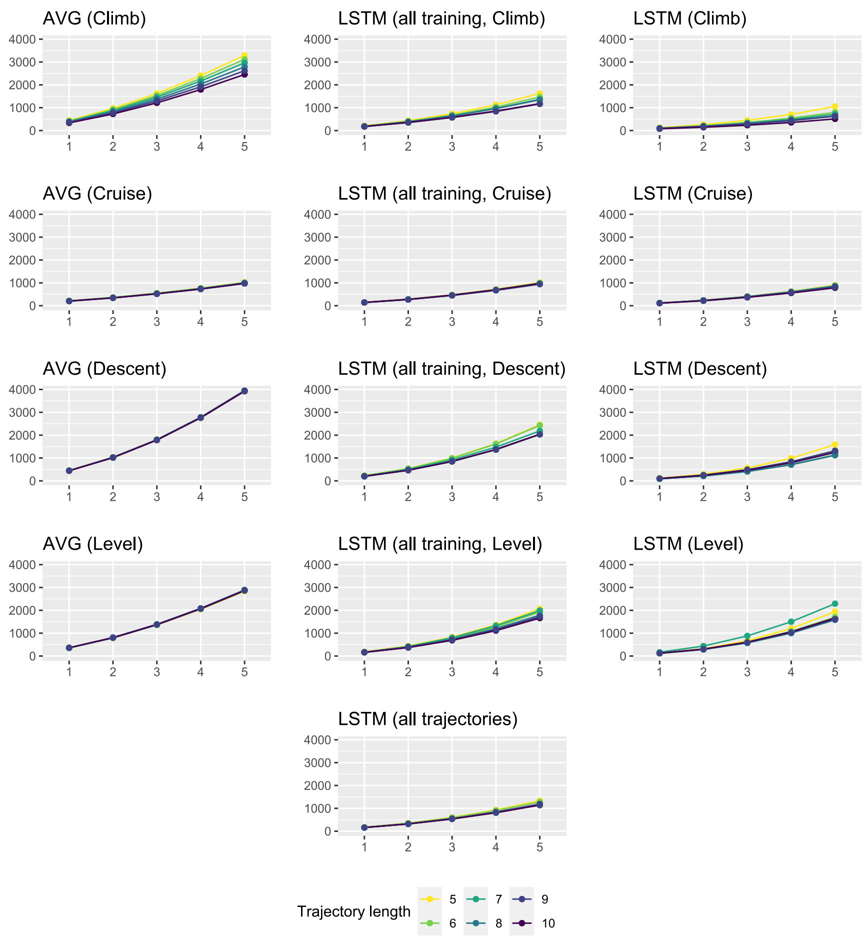


FIGURE 14. Results for S4. X-axis = prediction offset. Y-axis = MAE.

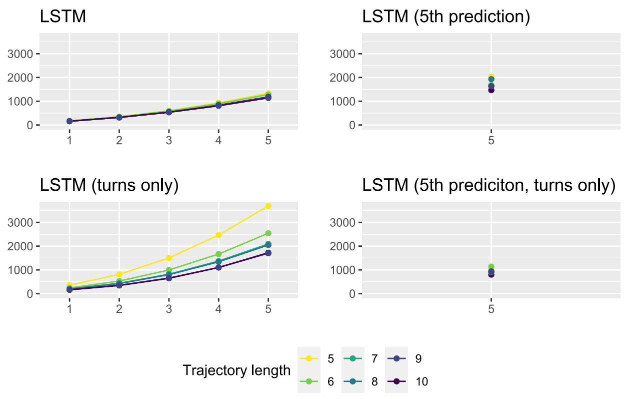


FIGURE 15. Results for S5. Delayed prediction results show a single point corresponding to the 5th prediction.

TABLE 8. Results summary for S5. X-axis = prediction offset. Y-axis = MAE.

Model	Error 1	Error 3	Error 5
LSTM	159.33	534.35	1140.61
LSTM (5th prediction)	-	-	1464.58
LSTM (turns only)	163.47	646.93	1723.07
LSTM (5th prediciton, turns only)	-	-	800.57

expanding the state vector features with extra ones, and adding trajectory-wide features. Regarding the extra state vector features, we add the true track (angle of the plane) and the vertical rate, both of which could be derived from the positions, but which could help the network by providing explicit information. Regarding trajectory-wide features, we add the latitude and longitude of the destination, the euclidean distance from the last known state vector to the destination, and the flight phase in said state vector. These features could improve performance by providing information about where the plane is leading towards, and at which point of the trajectory it is. To add these features to the model, we modified the LSTM model in Figure 4 so that trajectory-wide features are concatenated to the output of the LSTM 3 layer. We test these two strategies independently and jointly. Results are shown in Figure 17 and Table 9. Surprisingly, all additions lead to worse results. In the case of the extra state vector features (LSTM + extra), this is probably caused by how the model has more

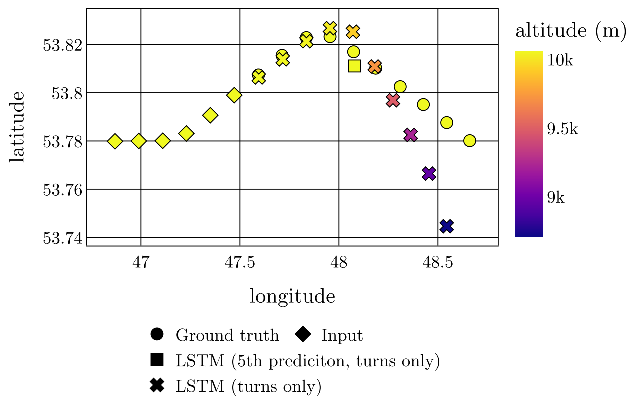


FIGURE 16. Example of delayed prediction for turning trajectory. Interactive 3D visualization.

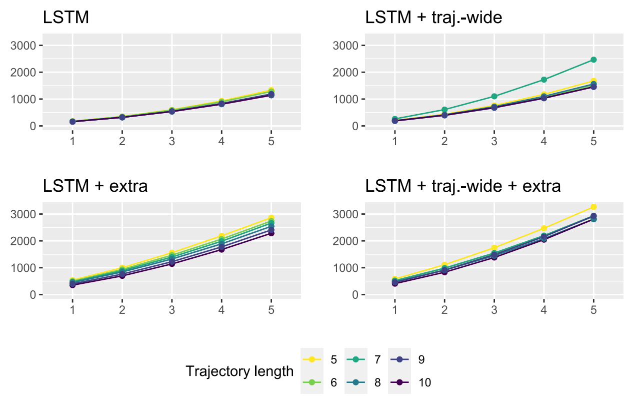


FIGURE 17. Results for S6. X-axis = prediction offset. Y-axis = MAE.

TABLE 9. Results summary for S6.

Model	Error 1	Error 3	Error 5
LSTM	159.33	534.35	1140.61
LSTM + traj.-wide	188.89	682.25	1455.89
LSTM + extra	351.62	1141.79	2283.27
LSTM + traj.-wide + extra	406.89	1381.05	2810.71

initial information, but is also forced to predict additional features for the consecutive state vectors. When it comes the trajectory-wide features, the worse results may be caused by the higher model complexity, and the additional features not providing crucial information for correctly predicting the most challenging cases.

E. ANSWERS TO THE RESEARCH QUESTIONS

Next, we give an explicit answer to the identified research questions from the results obtained in the experimental scenarios:

Q1: Do more complex architectures really achieve a significantly better performance?

The use of layers specialized in sequential inputs (e.g. LSTM layers) greatly improves results over simple dense networks. However, more elaborate architectures (CNN-LSTM) perform worse than simpler ones while being much more computationally expensive. Results from aforementioned complex architecture did not seem to clearly improve as the number of available trajectory points increased from 6 to 10, which leads us to conclude that these architectures would only make sense in a completely different scenario where the input contains hundreds of state vectors.

Q2: Do neural networks achieve better performance than simpler baselines that do not require training?

While the results obtained by neural networks were better overall than a simple baseline that approximately assumes the plane will keep moving in the same direction, the differences between both methods was relatively small compared to cost of training and applying a neural network. However, these results were expected, since the majority of a flight cruise takes place in a straight line. When predicting trajectories during turns or non-cruise phases, the differences were larger since the neural network is actually able to predict maneuvers to a certain extent.

Q3: To what extent does the length of the input trajectory (that is, the number of state vectors used to predict the next one) impact performance?

The use of the longer trajectories (10 state vectors) led to consistently better results than the shortest ones (5 state vectors). The differences were more significant in challenging scenarios: turning and non-cruise trajectories. Surprisingly, trajectory length had a lesser effect when performing delayed predictions in scenario S5.

Q4: When performing multi-step trajectory prediction, how does the prediction accuracy degenerate as we make more sequential predictions from the former ones?

In all scenarios accuracy degraded quickly beyond the first predictions, and as expected much faster in challenging scenarios. The use of specialized models helped to greatly reduce this error, but the MAE remains above 1000 for the fifth prediction. This may render neural networks unusable for performing a direct precise prediction (unless different techniques are applied to improve accuracy), but may be enough to predict expected flight intentions (e.g. to perform a turn).

Q5: To predict a distant position, is it better to perform multi-step prediction or delayed prediction?

While the use of delayed prediction was not beneficial for overall trajectories, it halved the MAE for turning trajectories, showing that specialized models for predictions in the far future may be a partial solution to the quickly degrading accuracy at the cost of flexibility. If such longer term predictions are needed, a reasonable option would be to train different models for predictions at fixed times, which would also be more efficient than repeatedly applying a model to perform multi-step prediction.

Q6: Is it better to use differential or absolute features?

Differential features resulted in results that were an order of magnitude better than absolute ones. While absolute features technically contain more information that may be useful for predictions with regards to the absolute position of a place, it is clear that this information should be included in addition to differential features.

Q7: How do models perform when trained with (and applied to) specific flight phases?

As expected, generic models that performed well in the overall experiments did so in part because of the predominance of the cruising phase during flights, being much less accurate during non-cruise phases. Training models for specific phases forced the models to focus on non-trivial cases and led to much lower MAE in all cases, with larger improvements in the climb and descent phases. The conclusions regarding this question, along with those from questions Q5 and Q8 seem to suggest that specialized models may be the key to improve the accuracy of predictions. Potential systems could create a variety of models for all different identifiable situations, which may correspond to different phases, aircrafts, weather conditions, etc.

A cost of doing so would be the need to gather further data that enables the training of models for specific situations.

Q8: How do models perform when trained with (and applied to) trajectories with less regular movement (more turns)?

Similarly to question Q7, general models are prone to error in non-straight trajectories, where specialized models are more successful. The difference with question Q7, however, is that unlike phases, it may be harder to determine when a plane is turning or is going to turn in the near future in order to apply the correct model. Models specialized in turns could be applied when a certain variance in direction has been detected, or when an additional model has predicted a flight will take place soon.

Q9: Does performance improve if additional trajectory-wide features related to the position of the plane in the flight are added?

Adding trajectory-wide features to the neural network architecture did not improve results; actually it made them slightly worse, probably because of the increased complexity of the network and potential overfitting. We conclude that leveraging these features may require a more complex model (avoiding over-complexity as mentioned in question Q1) or completely different features.

Q10: Does performance improve if we use extra state vector features in addition to positional ones (latitude, longitude, and altitude)?

Adding additional features beyond latitude, longitude and altitude led to significantly worse results, maybe for similar reasons to those mentioned in question Q9.

VI. CONCLUSION AND FUTURE WORK

We have presented a thorough study on the application of neural networks to short-term aircraft trajectory prediction. While the literature contains several implementations of the former, the experiments presented by them take place under very limited sets of conditions and overall do not address potential doubts about aspects that would be crucial when making decisions in a real-world environment. Specifically, we have identified ten open research questions that our study intends to answer. For that purpose, we have predicted trajectories from more than 800 flights, using several dozens of model variants and data configurations that, grouped and compared in six different scenarios, provide detailed insights into the research questions.

Our summarised findings regarding the aforementioned questions are the following: 1) a middle-complexity model achieved better results than a high complexity ones. 2) the performance of a neural network was better than that of a simpler baseline; while the later achieved similar results in straight trajectories, the difference was more pronounced in irregular trajectories. 3) the length of the input trajectory is particularly relevant when the plane is turning or in non-cruise phases. 4) as more predictions are made, the error increases exponentially, so that good accuracy can only be guaranteed for the first predictions; however specialised models can greatly diminish the rate at which the error increases. 5) delayed predictions seem to be particularly beneficial in irregular trajectories. 6) differential features are an order of magnitude better than normal ones. 7) training models for specific flight phases greatly diminishes the error of predictions, specially in non-cruise phases, where trajectories should be more irregular. 8) something similar happens when it comes to irregular trajectories: while a general model achieves good overall results, the error is much larger for cases with irregular movement. 9) the added trajectory-wide features didn't lead to improved results, but slightly worse ones. 10) adding features beyond latitude, longitude, and altitude led to significantly worse results.

Some of these conclusions seem to be in conflict with popular choices in the literature. For example, while differential features are always an order of magnitude better than normal ones, barely any of the works we have studied mentions them. The better results obtained with a model that specializes in making predictions for the far future (delayed-prediction) is also particularly interesting, since making predictions in the long term is much more challenging.

We hope that our explicit identification of these research questions will help guide future work so that parts of the efforts of the research community will be aimed towards further studying them. For example, new proposals could be evaluated with different trajectory sizes. Future work could also focus on answering single questions in greater detail; for example, a wider range of trajectory lengths could be tested, additional features could be added and fed to networks with specialised architectures, and delayed prediction could be evaluated under more challenging conditions (e.g. predicting further into the future). One of the most challenging scenarios to be tackled is the prediction of longer-term positions. Ways to tackle it could include the use of specialized models, or applying complex models to a large amount of input state vectors.

REFERENCES

[1] X. Fu and T. H. Oum, "Air transport liberalization and its effects on airline competition and traffic growth—An overview," in *The Economics of International Airline Transport*. Emerald Group Publishing Limited, 2014.

[2] Y. Q. Dong, "Implementing deep learning for comprehensive aircraft icing and actuator/sensor fault detection/identification," *Eng. Appl. Artif. Intell.*, vol. 83, no. 8, pp. 28–44, Aug. 2019.

[3] S. A. De Toledo, A. Anguera, J. Barreiro, J. Lara, and D. Lizcano, "A reinforcement learning model equipped with sensors for generating perception patterns: Implementation of a simulated air navigation system using ADS-B (automatic dependent surveillance-broadcast) technology," *Sensors*, vol. 17, no. 12, p. 188, Jan. 2017.

[4] Y. Wan, J. Tang, and S. Lao, "Research on the collision avoidance algorithm for fixed-wing UAVs based on maneuver coordination and planned trajectories prediction," *Appl. Sci.*, vol. 9, no. 4, p. 798, 2019.

[5] A. Degas, E. Kaddoum, M.-P. Gleizes, F. Adreit, and A. Rantrua, "Cooperative multi-agent model for collision avoidance applied to air traffic management," *Eng. Appl. Artif. Intell.*, vol. 102, Jun. 2021, Art. no. 104286.

[6] G. Di Gravio, M. Mancini, R. Patriarca, and F. Costantino, "Overall safety performance of air traffic management system: Forecasting and monitoring," *Saf. Sci.*, vol. 72, pp. 351–362, Feb. 2015.

[7] M. Schafer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up OpenSky: A large-scale ADS-B sensor network for research," in *Proc. 13th Int. Symp. Inf. Process. Sensor Netw.*, Apr. 2014, pp. 83–94.

[8] W. Zeng, X. Chu, Z. Xu, Y. Liu, and Z. Quan, "Aircraft 4D trajectory prediction in civil aviation: A review," *Aerospace*, vol. 9, no. 2, p. 91, Feb. 2022.

[9] R. Dalmau, M. Perez-Batlle, and X. Prats, "Real-time identification of guidance modes in aircraft descents using surveillace data," in *Proc. IEEE/AIAA 37th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2018, pp. 1–10.

[10] R. Rezaie and X. R. Li, "Trajectory modeling and prediction with waypoint information using a conditionally Markov sequence," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2018, pp. 486–493.

[11] Y. Lin, B. Yang, J. Zhang, and H. Liu, "Approach for 4-D trajectory management based on HMM and trajectory similarity," *J. Mar. Sci. Technol.*, vol. 27, no. 3, p. 7, 2019.

[12] C. E. Seah and I. Hwang, "Terminal-area aircraft tracking using hybrid estimation," *J. Guid., Control, Dyn.*, vol. 32, no. 3, pp. 836–849, May 2009.

[13] H.-C. Choi, C. Deng, and I. Hwang, "Hybrid machine learning and estimation-based flight trajectory prediction in terminal airspace," *IEEE Access*, vol. 9, pp. 151186–151197, 2021.

[14] J. Zhang, J. Liu, R. Hu, and H. Zhu, "Online four dimensional trajectory prediction method based on aircraft intent updating," *Aerosp. Sci. Technol.*, vol. 77, pp. 774–787, Jun. 2018.

[15] W. Schuster, "Trajectory prediction for future air traffic management—Complex manoeuvres and taxiing," *Aeronaut. J.*, vol. 119, no. 1212, pp. 121–143, Feb. 2015.

[16] A. Nuic, D. Poles, and V. Mouillet, "BADA: An advanced aircraft performance model for present and future ATM systems," *Int. J. Adapt. Control Signal Process.*, vol. 24, no. 10, pp. 850–866, Aug. 2010.

[17] F. Felix, M. Ruiz, C. Querejeta, E. Gallo, and J. Leones, "Predicting aircraft trajectory," U.S. Patent 9 250 099 B2, Feb. 2, 2016.

[18] R. Alligier, D. Gianazza, and N. Durand, "Learning the aircraft mass and thrust to improve the ground-based trajectory prediction of climbing flights," *Transp. Res. C, Emerg. Technol.*, vol. 36, pp. 45–60, Nov. 2013.

[19] M. G. Hamed, D. Gianazza, M. Serrurier, and N. Durand, "Statistical prediction of aircraft trajectory: Regression methods vs point-mass model," in *Proc. 10th USA/Eur. Air Traffic Manag. Res. Develop. Seminar*, 2013, pp. 630–639.

[20] S. T. Kanneganti, P. B. Chilson, and R. Huck, "Visualization and prediction of aircraft trajectory using ADS-B," in *Proc. NAECON-IEEE Nat. Aerosp. Electron. Conf.*, Jul. 2018, pp. 529–532.

[21] L. Zhao, Z. Li, A. Y. Al-Dubai, G. Min, J. Li, A. Hawbani, and A. Y. Zomaya, "A novel prediction-based temporal graph routing algorithm for software-defined vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 13275–13290, Aug. 2022.

[22] L. Zhao, T. Zheng, M. Lin, A. Hawbani, J. Shang, and C. Fan, "SPIDER: A social computing inspired predictive routing scheme for softwarized vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9466–9477, Jul. 2022.

[23] L. Zhao, Y. Liu, A. Y. Al-Dubai, A. Y. Zomaya, G. Min, and A. Hawbani, "A novel generation-adversarial-network-based vehicle trajectory prediction method for intelligent vehicular networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 2066–2077, Feb. 2021.

[24] Z.-J. Wu, S. Tian, and L. Ma, "A 4D trajectory prediction model based on the BP neural network," *J. Intell. Syst.*, vol. 29, no. 1, pp. 1545–1557, Aug. 2019.

[25] W. Min, W. Jiawei, G. Jinhui, S. Lihua, and A. Bogong, "Multi-point prediction of aircraft motion trajectory based on GA-Elman-Regularization neural network," *Integr. Ferroelectr.*, vol. 210, no. 1, pp. 116–127, Sep. 2020.

[26] Z. Shi, M. Xu, Q. Pan, B. Yan, and H. Zhang, "LSTM-based flight trajectory prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[27] Z. Xu, W. Zeng, X. Chu, and P. Cao, "Multi-aircraft trajectory collaborative prediction based on social long short-term memory network," *Aerospace*, vol. 8, no. 4, p. 115, Apr. 2021.

[28] W. Zeng, Z. Quan, Z. Zhao, C. Xie, and X. Lu, "A deep learning approach for aircraft trajectory prediction in terminal airspace," *IEEE Access*, vol. 8, pp. 151250–151266, 2020.

[29] K. Yang, M. Bi, Y. Liu, and Y. Zhang, "LSTM-based deep learning model for civil aircraft position and attitude prediction approach," in *Proc. Chin. Control Conf. (CCC)*, Jul. 2019, pp. 8689–8694.

[30] Z. Shi, M. Xu, and Q. Pan, "4-D flight trajectory prediction with constrained LSTM network," *IEEE Trans. Intell. Transp.*, vol. 22, no. 11, pp. 7242–7255, Nov. 2021.

[31] Z. Ma, M. Yao, T. Hong, and B. Li, "Aircraft surface trajectory prediction method based on LSTM with attenuated memory window," *J. Phys., Conf.*, vol. 1215, no. 1, May 2019, Art. no. 012003.

[32] L. Ma and S. Tian, "A hybrid CNN-LSTM model for aircraft 4D trajectory prediction," *IEEE Access*, vol. 8, pp. 134668–134680, 2020.

[33] P. G. Jeyasheeli, C. Kamaleshwar, and K. S. Aswin, "Deep learning methods for suicide prediction using audio classification," *J. Positive School Psychol.*, vol. 6, pp. 10479–10485, Jun. 2022.

[34] D. Ayala, A. Borrego, I. Hernández, and D. Ruiz, "A neural network for semantic labelling of structured information," *Exp. Syst. Appl.*, vol. 143, Apr. 2020, Art. no. 113053.

[35] G. Kostopoulos, M. Tsiakmaki, S. Kotsiantis, and O. Ragos, "Deep dense neural network for early prediction of failure-prone students," in *Machine Learning Paradigms: Advances in Deep Learning-Based Technological Applications*. Berlin, Germany: Springer, 2020, pp. 291–306.

[36] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," 2015, *arXiv:1511.08630*.

[37] E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller, "Multi-resolution linear prediction based features for audio onset detection with bidirectional LSTM neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 2164–2168.

[38] Y. Zhang, S. Qiao, S. Ji, and Y. Li, "DeepSite: Bidirectional LSTM and CNN models for predicting DNA-protein binding," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 4, pp. 841–851, Apr. 2020.

[39] Y. Bin, Y. Yang, F. Shen, N. Xie, H. T. Shen, and X. Li, "Describing video with attention-based bidirectional LSTM," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2631–2641, Jul. 2019.

[40] S. Ayadi and Z. Lachiri, "A combined CNN-LSTM network for audio emotion recognition using speech and song attributs," in *Proc. 6th Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, May 2022, pp. 1–6.

[41] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-based model to forecast stock prices," *Complexity*, vol. 2020, pp. 1–10, Nov. 2020.

[42] M. Cao, V. O. K. Li, and V. W. S. Chan, "A CNN-LSTM model for traffic speed prediction," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, May 2020, pp. 1–5.

[43] S. Bouaafia, R. Khemiri, F. E. Sayadi, M. Atri, and N. Liouane, "A deep CNN-LSTM framework for fast video coding," in *Proc. Int. Conf. Image Signal Process.* Cham, Switzerland: Springer, 2020, pp. 205–212.

[44] R. Ayala, D. Ayala, L. S. Vidal, and D. Ruiz, "OpenSkies—Integration of aviation data into the R ecosystem," *R J.*, vol. 13, no. 2, pp. 590–599, 2021.

[45] J. Sun, J. Ellerbroek, and J. Hoekstra, "Flight extraction and phase identification for large automatic dependent surveillance–broadcast datasets," *J. Aerosp. Inf. Syst.*, vol. 14, no. 10, pp. 566–572, Oct. 2017.

DANIEL AYALA is currently a Researcher and Professor with the University of Seville. His research with the DEAL Group, so far, has focused on techniques related to structured data integration and mining, and lately, in the field of knowledge graphs. His current research interests include the use of graph neural networks for attribute completion and gene-related predictions regarding antibiotic resistance.

He was a recipient of many awards, such as the Best Resource Paper Award in the ESWC 2019 Conference and the Best Paper of the Month awards from the University of Seville—ETSII.

RAFAEL AYALA received the B.Sc. degree in biochemistry from the University of Seville, in 2014, and the Ph.D. degree from Imperial College London, in 2020. He is currently a Postdoctoral Researcher with the Okinawa Institute of Science and Technology Graduate University, Japan. His Ph.D. focused on the analysis of the structure of biological macromolecules by cryo-EM, which is still one of his main research interests. His current research interests include the analysis of geospatial data, such as aircraft tracking and satellite positioning, as well as the application of machine learning techniques to these fields.

LARA SELLÉS VIDAL received the B.Sc. degree in medical biology from the University of Barcelona, in 2012, the M.Sc. degree in applied biosciences and biotechnology from Imperial College London, in 2013, and the Ph.D. degree in synthetic biology from the Centre for Synthetic Biology, Imperial College London, under the supervision of Dr. John Heap, in 2019. She is currently a Postdoctoral Fellow with the group of Prof. Yohei Yokobayashi, Japan Society for the Promotion of Science. Her research interests include the directed evolution of proteins, nucleic acids, and bio-production pathways, as well as the application of machine learning techniques to biological problems.

INMA HERNÁNDEZ has been the Coordinator of the master's program in software engineering (cloud, data, and IT management) with the Postgraduate School, University of Seville, since 2020. She is currently an Associate Professor with the University of Seville and a Founding Member of the Data Engineering Applications Laboratory. She is also a Principal Investigator for several projects funded by the Spanish National Research and Development Program. Her current research interests include data engineering and knowledge graphs. She has authored many peer-reviewed publications on these topics in top conferences and journals.

She is a very active reviewer and a member of several program committees at major conferences.

DAVID RUIZ is currently a Full Professor of software engineering with the University of Seville. He leads the Data Engineering Applications Laboratory, University of Seville, focusing his research on data engineering, knowledge graphs, and data integration. He has recently started two new related lines of research, focusing on the application of machine learning techniques for the automated retrieval and processing of aviation data, and for the genomic analysis of multi-resistant bacteria.