8-1-2023

A Data-Light and Trajectory-Based Machine Learning Approach for the Online Prediction of Flight Time of Arrival

Zhe Zheng
Nanjing University of Aeronautics and Astronautics

Bo Zou
College of Engineering

Wenbin Wei
San Jose State University, wenbin.wei@sjsu.edu

Wen Tian
Nanjing University of Aeronautics and Astronautics

Recommended Citation

Zhe Zheng, Bo Zou, Wenbin Wei, and Wen Tian. "A Data-Light and Trajectory-Based Machine Learning Approach for the Online Prediction of Flight Time of Arrival" *Aerospace* (2023). https://doi.org/10.3390/aerospace10080675 aerospace

MDPI

Article

A Data-Light and Trajectory-Based Machine Learning Approach for the Online Prediction of Flight Time of Arrival

Zhe Zheng [1,2,*], Bo Zou [3], Wenbin Wei [4] and Wen Tian [1,2]

[1] College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2] National Key Laboratory of Air Traffic Flow Management, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[3] Department of Civil, Materials, and Environmental Engineering, University of Illinois Chicago, Chicago, IL 60607, USA
[4] Department of Aviation and Technology, San Jose State University, San Jose, CA 95192, USA
* Correspondence: zhe_zheng@outlook.com

Abstract: The ability to accurately predict flight time of arrival in real time during a flight is critical to the efficiency and reliability of aviation system operations. This paper proposes a data-light and trajectory-based machine learning approach for the online prediction of estimated time of arrival at terminal airspace boundary (ETA_TAB) and estimated landing time (ELDT), while a flight is airborne. Rather than requiring a large volume of data on aircraft aerodynamics, en-route weather, and traffic, this approach uses only flight trajectory information on latitude, longitude, and speed. The approach consists of four modules: (a) reconstructing the sequence of trajectory points from the raw trajectory that has been flown, and identifying its best-matched historical trajectory which bears the most similarity; (b) predicting the remaining trajectory, based on what has been flown and the best-matched historical trajectory; this is achieved by developing a long short-term memory (LSTM) network trajectory prediction model; (c) predicting the ground speed of the flight along its predicted trajectory, iteratively using the current position and previous speed information; to this end, a gradient boosting machine (GBM) speed prediction model is developed; and (d) predicting ETA_TAB using trajectory and speed prediction from (b) and (c), and using ETA_TAB to further predict ELDT. Since LSTM and GBM models can be trained offline, online computation efforts are kept at a minimum. We apply this approach to real-world flights in the US. Based on our findings, the proposed approach yields better prediction performance than multiple alternative methods. The proposed approach is easy to implement, fast to perform, and effective in prediction, thus presenting an appeal to potential users, especially those interested in flight ETA prediction in real time but having limited data access.

Keywords: data-light and trajectory-based prediction; estimated time of arrival; trajectory reconstruction; matching; long short-term memory; gradient boosting machine

Citation: Zheng, Z.; Zou, B.; Wei, W.; Tian, W. A Data-Light and Trajectory-Based Machine Learning Approach for the Online Prediction of Flight Time of Arrival. *Aerospace* **2023**, *10*, 675. https://doi.org/10.3390/aerospace10080675

Academic Editor: Álvaro Rodríguez-Sanz

Received: 31 May 2023
Revised: 11 July 2023
Accepted: 25 July 2023
Published: 28 July 2023

1. Introduction

Air travel demand has experienced continuous growth over the past few decades despite a few temporary disruptions such as 9/11, the 2007–2009 economic recession, and the COVID-19 pandemic, leading to the increased density and complexity of air traffic and flight delays. With this context, the ability to accurately predict flight estimated time of arrival (ETA), especially in an online environment, is of critical importance to not only terminal airspace traffic control and management but also to airlines and airports for gate assignment, ground crew, and equipment deployment, and passenger pickup and connection services at the destination airport. Moreover, with the proliferation and popularity of air travel mobile apps, the need for a high-fidelity real-time prediction of flight time of arrival has grown substantially. While air navigation service providers (ANSPs) have access to rich information about airspace congestion, weather conditions, and air traffic control, which can be used to generate flight ETA, airlines, airport authorities, and air travel mobile app developers generally do not have access to such information. For airlines, they rely on flight plans and infrequent crew updates to generate their own ETA, which is less accurate than that generated using air traffic control [1,2]. Airports, when broadcasting flight arrival information, use the same ETA as is received from airlines (note that airport authorities are different from air traffic control at airports, which is part of an ANSP). Air travel mobile app developers mostly use data that are open-sourced or can be purchased at a reasonable price, such as from Flightradar24 [3]. Thus, developing an approach to predicting ETA that has a light use of easily accessible data will be useful and of interest to airlines, airports, and air travel mobile app developers.

This paper focuses on online ETA prediction while a flight is airborne. Two types of ETA are considered. The first type is ETA at the terminal airspace boundary, termed ETA_TAB. The second type is estimated landing time (ELDT), or time of touchdown. Dynamic and frequent updates of ETA while a flight is airborne are desirable to improve situation awareness and preparedness by potential users of the ETA information. As opposed to this, prior studies on trajectory-based ETA prediction involve the predictions of either when the flight arrives at or is within the terminal airspace boundary, or before the flight takes off. To serve the online prediction purpose, this study develops a machine learning approach that relies only on light, easily accessible data and is intended to provide a reasonably good online prediction of flight ETA while a flight is airborne, for intended users including airlines, airport authorities, and air travel mobile app developers, who do not have access to extensive real-time information such as en-route weather conditions but have needs for reasonably good ETA prediction.

To be more specific, the proposed data-light and trajectory-based machine learning approach only requires flight trajectory information, which can be easily accessed in real time from databases such as Flightradar24 [3]. This approach entails training long short-term memory (LSTM) neural networks for the prediction of trajectory point sequence and gradient boosting machine (GBM) for the prediction of flight ground speed along the predicted trajectory point sequence. LSTM, initially developed by Hochreiter and Schmidhuber [4], is an effective method for long-range sequential learning with numerous applications, including speech recognition [5–7], machine translation [8], and image captioning [9,10]. In our approach, historical flight trajectories are used to train LSTM models that are then used to iteratively predict the remaining trajectory of a flight under study. In doing so, a salient feature lies in introducing a construction layer that performs interval correction and trajectory prediction smoothing. For the prediction of flight ground speed, a GBM is employed, which builds on the idea of a sequential decision-tree-based ensemble. GBMs have multiple strengths and are also reported to have achieved significant success in performing prediction tasks [11–15]. The proposed approach is easy to implement and fast to perform. Our numerical results show that the prediction accuracy is better than that of several benchmark methods.

Next, we present a review of the existing literature on ETA prediction. Based on the review, our contributions are described in detail. The organization of the remainder of the paper is given last.

1.1. Literature Review

Multiple methods have been developed in the literature to predict flight ETA. An important stream of methods is based on flight trajectories. Trajectory-based methods can be grouped into two categories. The first category uses aircraft kinetic and kinematic models. Krozel et al. [16] develop a probabilistic model to determine an optimal flight path and ETA at the runway with a planning horizon of 20–30 min or 60–100 mile radius of the runway, by accounting for the fact that a flight often modifies its trajectory to avoid hazardous weather. Bai et al. [17] employ a kinematic trajectory generator to predict flight trajectory and air speed based on nominal speed profiles, to determine ETA at a downstream point. A probabilistic trajectory predictor for a multi-segment cruise when a flight is airborne is proposed in [18]. An averaged ground speed given wind uncertainties is used together with a nonparametric probabilistic transformation method to calculate flight time traversing each segment. Then, the total flight time is computed as the sum of flight times across all segments. Using kinematic and kinetic models, Zhang et al. [19] propose an online 4D trajectory prediction method to predict flight trajectories and ETA at waypoints inside the terminal airspace. Their method updates aircraft intent by calculating kinematic and geomatic constraints if a deviation occurs. Employing kinetic and kinematic models often requires a large number of data inputs, which may not be easily accessible or observable [20]. In addition, knowledge about aircraft kinetics and kinematics is needed for using and adapting such models, which can impose a challenge for some practitioners (e.g., airport authorities and air travel mobile app developers).

The second category of trajectory-based methods for ETA prediction is data-driven. Huang et al. [2] develop a probabilistic, hybrid linear system for predicting flight trajectory during the final approach (a planning horizon of around 8 nautical miles (nm) from landing) and ETA. The flight path is modeled as a state-dependent, discrete mode transition. Ayhan et al. [21] focus on predicting flight ETA before the flight actually departs. However, different from the study of Huang et al. [2], the trajectory is not part of the prediction. Instead, historical flight trajectories, along with a large number of airports, time, airline, aircraft type, weather, and air traffic congestion (both en-route and at airports) attributes, are used as input features to train several machine learning models. De Leege et al. [22] take actual flight trajectory data and meteorological data to train generalized linear models to predict flight trajectory and time of arrival at a significant point with a prediction horizon up to 45 nm from the runway. Wang et al. [23] introduce a two-part model which first performs cluster-based preprocessing and then multicell neural network-based prediction using Automatic Dependent Surveillance–Broadcast (ADS-B) data including aircraft position, heading, and horizontal and vertical speed, for short-term trajectory prediction in the terminal maneuvering area. Using similar ADS-B data, Wang et al. [24] extend the previous work by clustering flights based on runway-in-use data and introducing an ensemble learning strategy called stacking to predict flight ETA, when predictions are made at the entry point of the terminal maneuvering area. Overall, these trajectory-based methods typically require a significant volume of data (such as en-route weather information) compared with our proposed approach. In addition, the predictions are performed either when a flight arrives at or is within the terminal airspace boundary, or before a flight takes off. These methods do not deal with more general predictions when a flight is anywhere while airborne.

Apart from the above trajectory-based ETA prediction studies, other methods have also been considered in the literature. Glina et al. [25] propose quantile regression forests, a tree-based ensemble method, to estimate flight landing time while the flight is within 60 nm of the destination airport. In this method, flight trajectory data, time of day, weather conditions, and runway availability are considered. Kim [26] applies spline smoothing-based nonparametric additive techniques to predict flight ETA at the time when a flight departs, considering departure delay, scheduled airborne time, weather conditions at the destination airport, time of day, day of the month, and month of the year as predictor variables. Achenbach and Spinler [27] predict flight ETA at the point of gate departure, using an ensemble of GBM and linear regression as the prediction method. In total, 71 predictor variables encompassing flight data, weather forecasts, and departure and arrival airport congestion are included. Lui et al. [28] predict the flight time inside the terminal maneuvering area before landing using random forest with flight altitude, ground speed, vertical rate, heading angle at the terminal control area entry point, and time of day as input features. Again, these methods consider prediction either when a flight is very close to the destination airport or prior to departure. The predictions are not dynamic and are performed at specific time or location points.

1.2. Contributions of the Paper

In view of the existing literature, the contribution of this paper lies in proposing a new data-light approach to real-time ETA prediction while a flight is airborne. Our approach requires light data as only the flown trajectory of the flight under study and historical trajectories of similar flights are needed. The fundamental idea is to perform separate and iterative predictions of flight positions and corresponding ground speeds using LSTM and GBM for the remaining trajectory of a flight while airborne. Based on the predicted flight positions and speeds, flight ETA_TAB and ELDT are computed. As LSTM and GBM models can be trained offline, the prediction takes small online computation efforts and can be performed in real time.

In performing the ETA prediction, two salient features, namely trajectory points reconstruction and a smoothing procedure, are designed and incorporated into the proposed approach. For trajectory points' reconstruction, we recognize that raw trajectory points are unevenly distributed and reconstruct trajectory point sequences to be of equal distance intervals, to enable effective LSTM prediction, instead of with equal time intervals [20,29–31]. The implementation of the smoothing procedure is motivated by the understanding that the LSTM-predicted flight trajectory may not always be smooth. To mitigate this, we propose to smooth the LSTM-predicted trajectory using a historic trajectory with the most similarity. The smoothing procedure is embedded in a tailored construction layer in LSTM prediction.

1.3. Organization of the Paper

The remainder of the paper is organized as follows: Section 2 presents the methodology, starting with the definition of a flight trajectory, the overall methodology framework, and its component modules. Section 3 shows the implementation of the approach using real-world flight data, with a detailed demonstration of flight trajectory matching, trajectory prediction, speed prediction, and ETA_TAB and ELDT prediction. The results on a larger number of historical flights are also presented, including a comparison with alternative approaches. Section 4 summarizes the paper, offers further discussions of the proposed approach, and suggests possible directions for extending the research.

2. Methodology

2.1. Definition

To perform flight trajectory prediction, it is necessary to define a flight trajectory. As aircraft ground speed is used for predicting ETA_TAB and ELDT, altitude information is not needed in characterizing a flight trajectory. Instead, a flight trajectory is represented by a sequence of points each specified by longitude, latitude, and ground speed, as defined below.

Definition 1: *Flight trajectory. A flight trajectory consists of a sequence of points: $TP = [p_1, p_2, \ldots, p_R]$, where $p_i = \left[\lambda_i, \varphi_i, v_g^i\right]$ denotes longitude λ_i, latitude φ_i, and ground speed v_g^i at point i. R is the total number of points in the trajectory.*

Information on p_i for the numerical implementation of the proposed approach will be obtained from Flightradar24 (Section 3.1 describes more details). At each point on a flight trajectory, we further introduce $d_{\text{remain}}^{\text{GCD}}$, the remaining great-circle distance (GCD) to the destination airport. $d_{\text{remain}}^{\text{GCD}}$ is calculated based on (λ, φ) of the trajectory point and the longitude and latitude of the destination airport, which are obtained from the OpenFlights database [32].

2.2. Overall Framework for ETA_TAB and ELDT Prediction

As mentioned earlier, when a flight is airborne, ETA_TAB is calculated by adding an estimate of the remaining en-route time up to the terminal airspace boundary, to the current time. Once ETA_TAB is obtained, ELDT is calculated by further adding an estimated terminal approach time, i.e., the time between when the flight enters the terminal airspace and touchdown. The remaining en-route time up to the terminal airspace boundary is estimated using the predicted remaining flight trajectory and ground speed at predicted trajectory points. As ground speed is horizontal speed relative to the ground, a flight trajectory only concerns the horizontal part of the trajectory. During the terminal approach, a flight's ground speed varies more drastically and irregularly depending on traffic density and control, weather, and other factors. As such, it will be more difficult to accurately predict the terminal arrival route of a flight. In view of this, in the paper, we estimated the terminal approach time through the sampling of historical terminal approach times.

Our proposed approach consists of four modules: (a) reconstructing the sequence of trajectory points from the raw trajectory that has been flown and identifying its best-matched historical trajectory that bears the most similarity; (b) predicting the remaining trajectory, based on what has been flown and the best-matched historical trajectory, which is achieved by developing an LSTM trajectory prediction model; (c) predicting ground speed of the flight along its predicted trajectory, iteratively using current position and previous speed information, and for this purpose, a GBM speed prediction model is developed; and (d) predicting ETA_TAB using trajectory and speed prediction from (b) and (c), and using ETA_TAB to further predict ELDT. The connections of the four modules are shown in Figure 1.

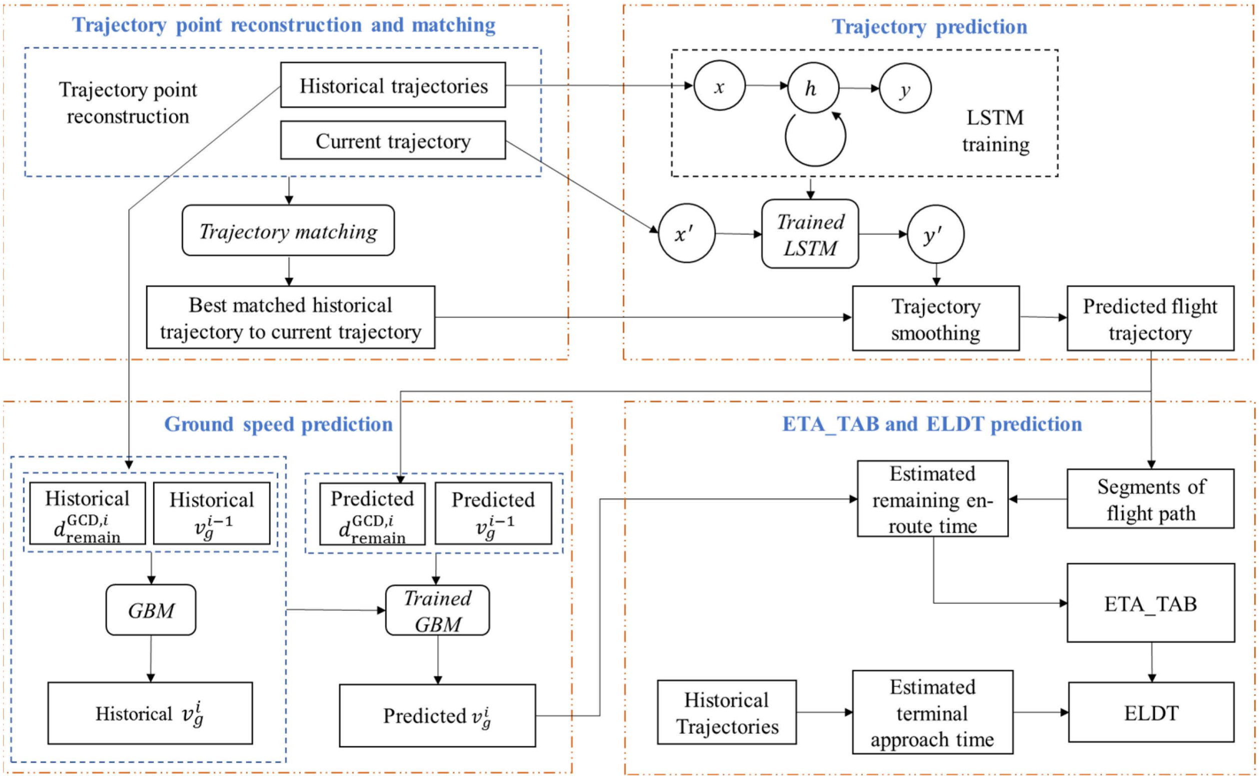

Trajectory point reconstruction and matching
Trajectory point reconstruction
Historical trajectories
Current trajectory
Trajectory matching
Best matched historical trajectory to current trajectory
Trajectory prediction
x
h
y
LSTM training
x'
Trained LSTM
y'
Trajectory smoothing
Predicted flight trajectory
Ground speed prediction
Historical $d_{\text{remain}}^{\text{GCD},i}$
Historical v_g^{i-1}
GBM
Historical v_g^i
Predicted $d_{\text{remain}}^{\text{GCD},i}$
Predicted v_g^{i-1}
Trained GBM
Predicted v_g^i
ETA_TAB and ELDT prediction
Estimated remaining en-route time
Segments of flight path
ETA_TAB
Historical Trajectories
Estimated terminal approach time
ELDT

Figure 1. The overall framework for ETA_TAB and ELDT prediction.

A few notations are worth mentioning in Figure 1. For trajectory prediction, x and y represent the input and output sequences of the historical trajectories. h denotes the hidden layer of the LSTM neural networks for trajectory prediction. x' and y' are the input and output sequences of the current trajectory. v_g^{i-1} and v_g^i indicate flight ground speed in the $(i-1)$th and ith steps in the distance sequence of a flight's trajectory.

2.3. *Trajectory Point Reconstruction and Matching*

2.3.1. Trajectory Point Reconstruction

The raw flight trajectories were obtained from Flightradar24, which come from ADS-B data and consist of a sequence of trajectory points each with longitude, latitude, and time information [3]. The raw trajectory points are of unequal time or distance intervals, making them unsuitable for LSTM modeling. Different from the previous LSTM studies for flight trajectory prediction that are based on using/constructing a sequence of trajectory points of equal time intervals, we opted for reconstructing a sequence of trajectory points of equal GCD. In Figure 2, the upper half presents the raw trajectory points, which are unevenly spaced. We connected each pair of neighboring trajectory points using a straight line. The lines form an approximate trajectory of the flight. Then, starting from the first trajectory point, we placed a sequence of points that are a nm apart along the continuous trajectory. In this paper, we considered $a = 1$ nm. (Considering that 1 nm is very small compared with the length of a flight trajectory (which is hundreds or thousands of miles), any residual after taking the integer value of the total trajectory length, which is less than 1 nm, can be reasonably neglected.) In this way, the sequence of reconstructed trajectory points was obtained, as shown in the lower half of Figure 2.

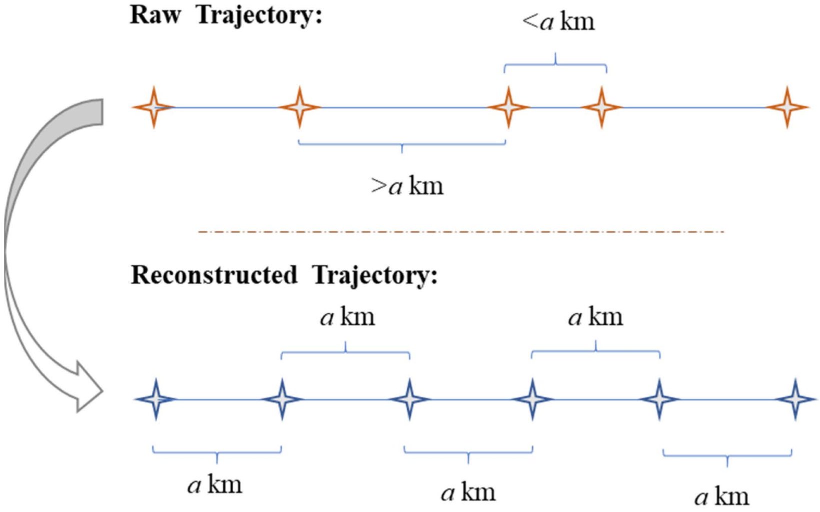


Figure 2. Flight trajectory before and after construction.

It is worth noting that, compared with other methods in which trajectory points are reconstructed with equal time intervals, a notable advantage of our approach is that the determination of trajectory points is purely based on distance. No time or speed information is required. By contrast, to identify the next reconstructed trajectory point (in terms of latitude and longitude) with a given time separation from the current reconstructed trajectory point, flight speed information would be needed in addition to latitude/longitude information of the current reconstructed trajectory point. However, precise information on flight speed between the two trajectory points is not available to us. A workaround could be taking an average speed between two consecutive raw trajectory points, to approximate the true speed along the flight trajectory. However, doing so would incur errors. For example, the approximation would not be able to capture speed variations between any two raw trajectory points, consequently generating and propagating errors in reconstructing trajectory points.

2.3.2. Trajectory Matching

The objective of trajectory matching is to identify a flight trajectory among historical trajectories of the same airport pair and flight number that is most similar to the trajectory of the current flight. The best-matched historical trajectory is then used in a smoothing procedure while performing trajectory prediction. The rationale is that if the flight under study—based on what has been flown—follows a very similar trajectory pattern as a historical flight, then pilots on the two flights are likely to have similar intents in flying. Consequently, the remaining part of the flight under study may also bear a close similarity to that historical flight. Note that both historical and current trajectories were reconstructed following Section 2.3.1 before matching was performed.

To be sure, one might think of estimating ETA_TAB and ELDT based on flight plans. However, very detailed flight plans were not available to us, nor are they available to potential non-airline users of the proposed approach, such as airport authorities and air travel mobile app developers. Moreover, a flight plan only represents the planned airborne route. The actual flight path may deviate—sometimes considerably—from the flight plan due to unexpected en-route traffic congestion and weather conditions. Thus, predicting the flight trajectory based on the flight plan could lead to inaccurate estimates (our benchmarking results will show this; see Section 3.5.2). On the other hand, unlike relying on flight plans, flight path deviations due to unexpected situations are implicitly reflected in the realized flight trajectories (both historical and current flights) and thus can be taken into account by matching the current flight's trajectory with the most similar historical flight trajectory and learning the common flying behavior from the latter trajectory.

To identify the most similar (best-matched) historical flight trajectory, a similarity score was introduced to quantify the similarity between two trajectories. Figure 3 shows how the similarity score of trajectory 1 (T_1), which corresponds to the current flight, with a historical trajectory T_2. In the figure, T_1 has five trajectory points: $T_1 = [p_1^1, p_2^1, p_3^1, p_4^1, p_5^1]$. Similarly, $T_2 = [p_1^2, p_2^2, p_3^2, p_4^2, p_5^2]$. For each trajectory point on T_1, we check if there exists any point on trajectory T_2 that is close enough. To illustrate, p_2^1 is a trajectory point on T_1. To determine if point p_2^1 is close enough to T_2, GCD between p_2^1 and each trajectory point on T_2 is computed, denoted by $d_1, d_2, \ldots, d_5$, as shown in Figure 3. If there exists an $i = 1, 2, \ldots, 5$ such that $d_i \leq \varepsilon$, where ε is the threshold for measuring closeness, then p_2^1 is considered close enough to T_2. All points on T_1 are checked in this way. The similarity score of T_2 to T_1, $S_{T_2 \to T_1}$, is calculated as follows:

$$S_{T_2 \to T_1} = \frac{N_{T_2 \to T_1}}{N_{T_1}} \tag{1}$$

where $N_{T_2 \to T_1}$ is the number of points on T_1 that are identified close enough to T_2. N_{T_1} is the total number of points on T_1. For example, if three points, p_1^1, p_2^1, and p_3^1, on T_1 are close enough to T_2, then $N_{T_2 \to T_1} = 3$. Given that $N_{T_1} = 5$, $S_{T_2 \to T_1} = \frac{3}{5} = 0.6$.

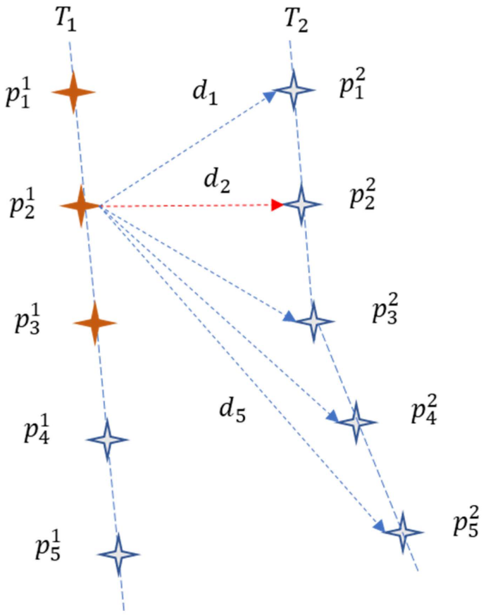


Figure 3. Illustration of how to identify close enough points on a historical flight trajectory (T_2) when calculating the similarity score of trajectory T_2 to trajectory T_1.

Once similarity scores of all historical flight trajectories of the same airport pair and flight number to the current flight trajectory are obtained, the trajectory with the highest similarity score is identified as the best-matched historical trajectory. Based on the above description, the overall computation complexity of trajectory matching is $O(mnk + k\log k)$, where m is the number of trajectory points of the current flight, n is the maximum number of trajectory points on a historical trajectory, and k is the number of historical trajectories. The first part in $O(\cdot)$ pertains to distance calculation, while the second part in $O(\cdot)$ relates to identifying the historical trajectory with the maximum similarity score, which requires sorting the similarity scores of all k historical trajectories.

2.4. Trajectory Prediction

2.4.1. LSTM Neural Network

The LSTM neural network model was employed to predict the remaining trajectory of the current flight. LSTM neural networks are a specific implementation of recurrent neural networks (RNNs) capable of learning the order dependence in sequence (Hochreiter and Schmidhuber, 1997) [4]. Compared with RNNs, LSTM networks have the advantage of overcoming the drawbacks of long sequence remembering and gradient explosion or vanishing during training. In this section, we describe the structure and mechanisms of general LSTM networks, based on which we discuss how LSTM is adapted and refined for trajectory prediction in Section 2.4.2.

LSTM networks have a chain-like structure with repeating blocks, as shown in Figure 4. The internal structure of an LSTM block is specified in the middle block. LSTM can remove or add information to the cell state (C_t), which is key to LSTM and runs through the entire chain, by structures known as gates, which selectively allow information to pass through. An LSTM block has three gates with the sigmoid activation function to control the cell state. More specifically, the sigmoid function is used to output a value in the range between 0 and 1 and describe the amount of information that passes through.

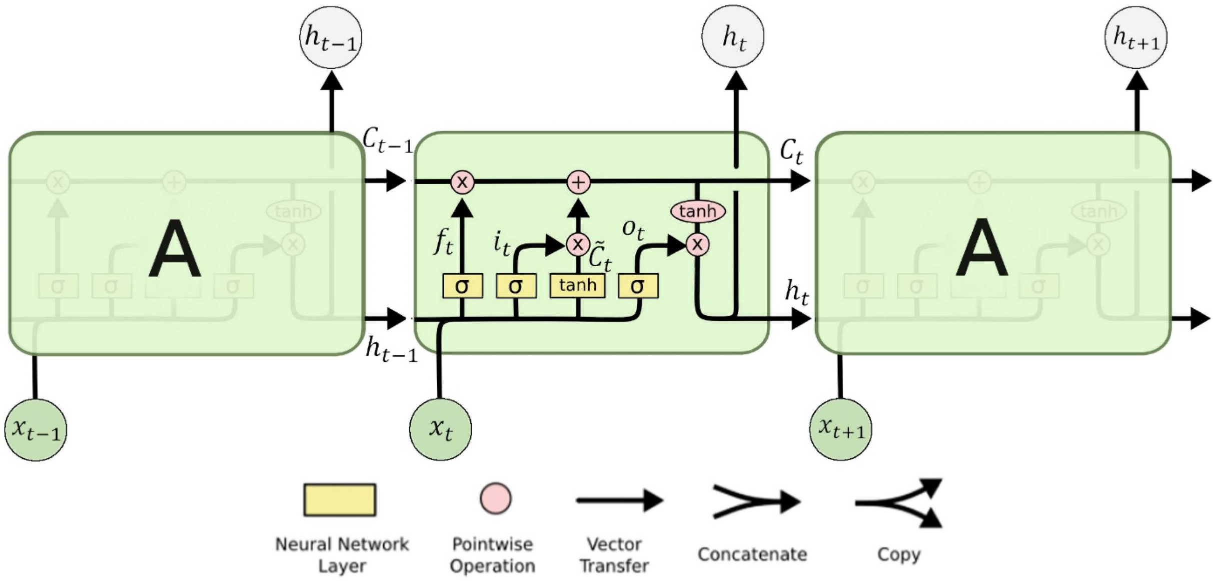


Figure 4. The architecture of LSTM neural networks [33].

The data fed to the LSTM gates include input at the current step, x_t, and the hidden state of the previous step, h_{t-1}. In flight trajectory prediction, a step corresponds to a trajectory point. x_t and h_{t-1} are processed to compute the values of the three gates: input, forget, and output, denoted by i_t, f_t, and o_t, as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4}$$

where σ is the sigmoid function. W_i, W_f, and W_o are weight parameters. b_i, b_f, and b_o are bias parameters.

In Figure 4, a candidate cell state $\tilde{C}_t$ is introduced for the calculation of cell state C_t. The computation of $\tilde{C}_t$ is similar to that of the three gates, with a hyperbolic tangent (tanh) function ranging from -1 to 1 used as the activation function:

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{5}$$

where W_c is a weight parameter; b_c is a bias parameter.

To calculate C_t, four variables are involved, as shown in Equation (6). In the equation, the forget gate f_t defines the amount of the past cell state C_{t-1} to be retained. The input gate i_t governs how the new data are considered via $\tilde{C}_t$.

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \tag{6}$$

where $\circ$ is the Hadamard product, for which the elements corresponding to the same rows and columns of two matrices are multiplied to form a new matrix. If f_t and i_t are, respectively, one and zero, the previous cell state C_{t-1} would be saved and passed to C_t.

With the three gates and the cell state introduced, the hidden state h_t is defined using Equation (7). The tanh value of the cell state ensures that the value of h_t is between -1 and 1. If the output gate o_t equals one, then all the information of C_t is passed to h_t. In contrast, an h_t value of zero indicates that no information of C_t is passed into the next step as the hidden state value.

$$h_t = o_t \times \tanh(C_t) \tag{7}$$

The above description is for a single LSTM block. In practice, multiple LSTM blocks are stacked to enable learning more complex patterns from sequence data.

2.4.2. LSTM Network for Trajectory Prediction

We adopted LSTM for flight trajectory prediction in the following way: Our LSTM network had two stacked LSTM layers with 64 hidden units with five inputs and one output. The batch size in training was 10. Each input consisted of three features describing the flight state at a given position: longitude, latitude, and the remaining GCD from the position to the destination airport, denoted by $d^{\text{GCD}}_{\text{remain}}$. Each of these three features was normalized using min–max scaling with the minimum and maximum taking from all historical flights of the same airport pair and flight number so that the feature value would always be between 0 and 1. Although in principle, $d^{\text{GCD}}_{\text{remain}}$ can be calculated using longitude and latitude, our experiments reveal that explicitly including $d^{\text{GCD}}_{\text{remain}}$ improves the prediction accuracy of the trained LSTM model. A possible reason for this is that the sequence of $d^{\text{GCD}}_{\text{remain}}$ along a flight's trajectory is expected to follow a decreasing trend. As a result, explicitly having $d^{\text{GCD}}_{\text{remain}}$ helps guide the trajectory prediction in the direction of reducing $d^{\text{GCD}}_{\text{remain}}$. Figure 5a illustrates the process of LSTM training, where x denotes the input and y denotes the output. Subscript c denotes "current". A dropout layer was added with a dropout rate of 0.5 between the two LSTM layers to reduce overfitting.

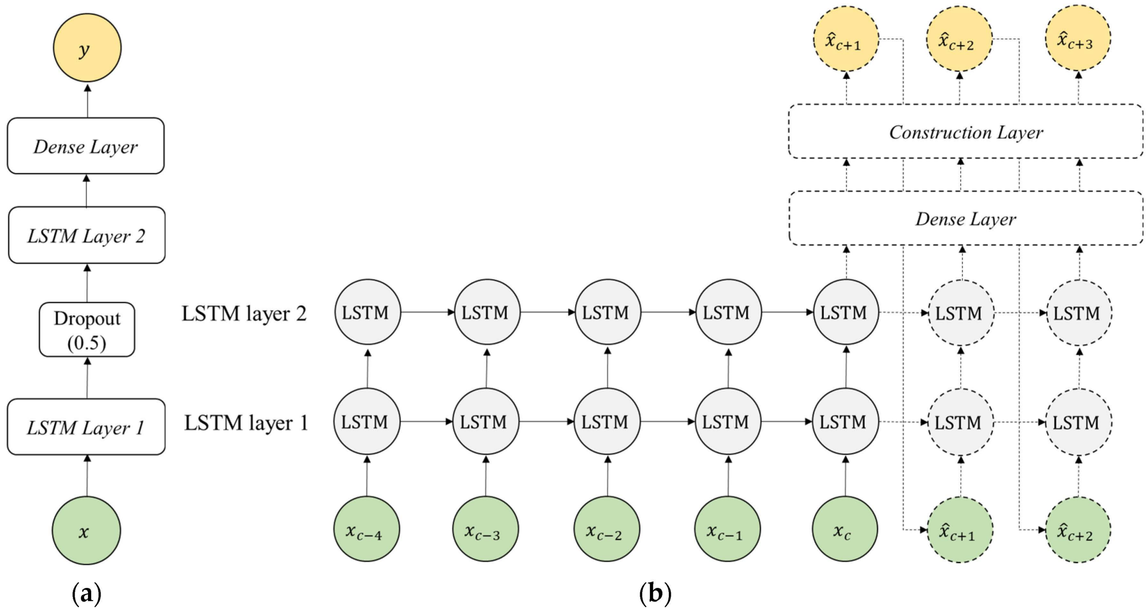


Figure 5. The architecture of the proposed trajectory prediction approach using LSTM: (**a**) training; (**b**) inference.

Once an LSTM model was trained using the historical trajectories, a one-step-ahead strategy was adopted to perform trajectory point prediction for the current flight's trajectory. To illustrate, consider the current trajectory point c. Then, $x_c, x_{c-1}, x_{c-2}, x_{c-3}, x_{c-4}$ are used by the LSTM model to generate the prediction of x_{c+1}, denoted by $\hat{x}_{c+1}$. As mentioned above, each $x_c = (\lambda_c, \varphi_c, d^{\text{GCD}}_{c,\text{remain}})$, where $\lambda_c, \varphi_c, d^{\text{GCD}}_{c,\text{remain}}$ denotes longitude, latitude, and the remaining distance to the destination airport. The prediction of x_{c+1} involves feeding the output of the last LSTM block of hidden layer 2 to a dense layer and a construction layer. The construction layer is used to adjust the predicted position of trajectory point $c+1$, so that the predicted point is smoothed and a nm from trajectory point c, where $a = 1$ nm is the prespecified separation distance (see Section 2.3.1). Once $\hat{x}_{c+1}$ is generated, $\hat{x}_{c+1}$ is used together with $x_c, x_{c-1}, x_{c-2}, x_{c-3}$ to generate $\hat{x}_{c+2}$. $\hat{x}_{c+2}$ and $\hat{x}_{c+1}$ along with x_c, x_{c-1}, x_{c-2} are further used to generate $\hat{x}_{c+3}$. This iterative process continued until reaching the terminal airspace boundary of the destination airport.

The construction layer consisted of four procedural steps, as shown in Figure 6. Suppose the output of the dense layer is $\hat{x}^0_{c+1} = (\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}, \hat{d}^{\text{GCD},0}_{c+1,\text{remain}})$. In the first step, an interval correction procedure was employed to adjust $\hat{\lambda}^0_{c+1}$ and $\hat{\varphi}^0_{c+1}$ so to have the GCD between the current position (λ_c, φ_c) and the adjusted prediction position, denoted by $(\hat{\lambda}^1_{c+1}, \hat{\varphi}^1_{c+1})$, equal to $a = 1$ nm. In the second step, we constructed a sub-trajectory of the best-matched historical trajectory obtained in Section 2.3 for the purpose of trajectory prediction smoothing. To illustrate how this is achieved, let us use $T^b = [p^b_1, p^b_2, p^b_3, \ldots]$ to denote the reconstructed trajectory point sequence of the best-matched historical trajectory (we use superscript b to represent the "best-matched trajectory" in this paper). We "virtually" insert (λ_c, φ_c) into the point sequence of T^b, in the position right after the point closest to (λ_c, φ_c). The sub-trajectory is an artificial trajectory starting from (λ_c, φ_c) and connecting to the subsequent points in the sequence till the end of T^b. To ensure equal-distance separation, the points are further reconstructed following the same procedure in Section 2.3.1, so that starting from (λ_c, φ_c), the subsequent reconstructed points are $a = 1$ nm apart. The resulting sub-trajectory is denoted by $T^{b,r}_c = [p^{b,r}_0, p^{b,r}_1, p^{b,r}_2, \ldots]$ where $p^{b,r}_0 = (\lambda_c, \varphi_c)$ and $p^{b,r}_k = (\lambda^{b,r}_{c+k}, \varphi^{b,r}_{c+k})$ for $k = 1, 2, \ldots$ (subscript b means the best-matched trajectory; subscript r means the remaining trajectory after the current position). In the third step, a smoothing procedure using $T^{b,r}_c$ was applied to $(\hat{\lambda}^1_{c+1}, \hat{\varphi}^1_{c+1})$, leading to updated longitude and latitude $\left(\hat{\lambda}^2_{c+1}, \hat{\varphi}^2_{c+1}\right)$. Finally, in the fourth step, the interval correction procedure was again employed, now to $\left(\hat{\lambda}^2_{c+1}, \hat{\varphi}^2_{c+1}\right)$, to ensure that the predicted point $c+1$ is $a = 1$ nm apart from point c after smoothing. The output is $\left(\hat{\lambda}_{c+1}, \hat{\varphi}_{c+1}\right)$. Based on $\left(\hat{\lambda}_{c+1}, \hat{\varphi}_{c+1}\right)$, $\hat{d}^{\mathrm{GCD}}_{c+1,\mathrm{remain}}$ is also calculated. Thus, $\hat{x}_{c+1} = \left(\hat{\lambda}_{c+1}, \hat{\varphi}_{c+1}, \hat{d}^{\mathrm{GCD}}_{c+1,\mathrm{remain}}\right)$ is generated. Below, we describe the interval correction and smoothing procedures in detail.

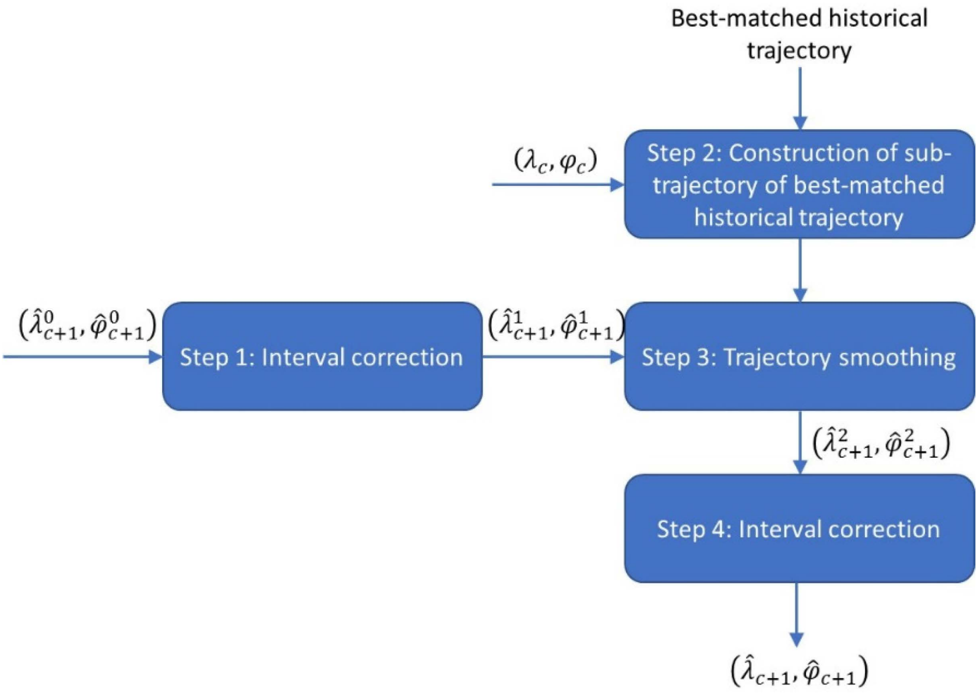

Best-matched historical trajectory
(λ_c, φ_c)
Step 2: Construction of sub-trajectory of best-matched historical trajectory
$\left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}\right)$
Step 1: Interval correction
$\left(\hat{\lambda}^1_{c+1}, \hat{\varphi}^1_{c+1}\right)$
Step 3: Trajectory smoothing
$\left(\hat{\lambda}^2_{c+1}, \hat{\varphi}^2_{c+1}\right)$
Step 4: Interval correction
$\left(\hat{\lambda}_{c+1}, \hat{\varphi}_{c+1}\right)$

Figure 6. Illustration of the four steps performed in the construction layer to generate $\left(\hat{\lambda}_{c+1}, \hat{\varphi}_{c+1}\right)$ using $\left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}\right)$ from the dense layer, (λ_c, φ_c), and the best-matched historical trajectory.

The interval correction procedure works as follows: Again, consider the output of the dense layer to be $\hat{x}^0_{c+1} = \left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}, \hat{d}^{\mathrm{GCD},0}_{c+1,\mathrm{remain}}\right)$. As the GCD between (λ_c, φ_c) and $\left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}\right)$ may not be the prespecified separation distance of $a = 1$ nm, the interval correction procedure makes an adjustment by identifying the trajectory point that is $a = 1$ nm from (λ_c, φ_c) toward $\left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}\right)$. To this end, we first identify the tracking angle θ clockwise from true north between (λ_c, φ_c) and $\left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}\right)$ (Figure 7).

$$\theta = \arctan\left(\frac{\sin\left(\hat{\lambda}^0_{c+1} - \lambda_c\right) \cdot \cos \hat{\varphi}^0_{c+1}}{\cos \varphi_c \cdot \sin \hat{\varphi}^0_{c+1} - \sin \varphi_c \cdot \cos \hat{\varphi}^0_{c+1} \cdot \cos\left(\hat{\lambda}^0_{c+1} - \lambda_c\right)}\right) \tag{8}$$

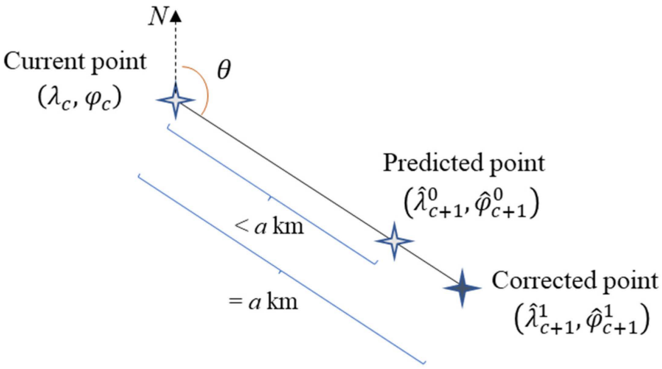

N
θ
Current point
(λ_c, φ_c)
Predicted point
$\left(\hat{\lambda}^0_{c+1}, \hat{\varphi}^0_{c+1}\right)$
$< a$ km
$= a$ km
Corrected point
$\left(\hat{\lambda}^1_{c+1}, \hat{\varphi}^1_{c+1}\right)$

Figure 7. Illustration of the interval correction procedure.

Keeping the tracking angle θ fixed, the longitude and latitude of the corrected point $\left(\hat{\lambda}^1_{c+1}, \hat{\varphi}^1_{c+1}\right)$ are calculated using Equations (9)–(11) [34], where $R = 6371$ km is the Earth's radius.

$$\delta = \frac{a}{R} \tag{9}$$

$$\hat{\varphi}^1_{c+1} = \arcsin(\sin\varphi_c \cdot \cos\delta + \cos\varphi_c \cdot \sin\delta \cdot \cos\theta) \tag{10}$$

$$\hat{\lambda}^1_{c+1} = \lambda_c + \arctan\left(\frac{\sin\theta \cdot \sin\delta \cdot \cos\varphi_c}{\cos\delta - \sin\varphi_c \cdot \sin\hat{\varphi}^1_{c+1}}\right) \tag{11}$$

For the smoothing procedure, as the LSTM model was trained based on a number of historical trajectories, our experiments show that some "zigzags" can occur to the predicted trajectories, which is not consistent with the real-world flying behavior. To mitigate this, we employed a procedure through which the LSTM-predicted trajectory is smoothed with the sub-trajectory of the best-matched historical trajectory. Specifically, for the kth predicted trajectory point from the current flight position c, the smoothed longitude and latitude $(\hat{\lambda}^2_{c+k}, \hat{\varphi}^2_{c+k})$ are calculated using Equations (12)–(14). In Equation (12), α_{c+k} is the smoothing coefficient as the ratio of GCDs from the kth predicted trajectory point and from the current flight position to the destination airport ($d^{\text{GCD}}_{c+k,\text{remain}}/d^{\text{GCD}}_{c,\text{remain}}$). The farther the prediction is (i.e., larger k), the smaller the value of $d^{\text{GCD}}_{c+k,\text{remain}}$ and consequently α_{c+k}. With α_{c+k}, the smoothed longitude of the kth predicted trajectory point is $\alpha_{c+k}\hat{\lambda}^1_{c+k} + (1-\alpha_{c+k})\lambda^{b,r}_{c+k}$. The rationale is that when the LSTM prediction gets farther into the future, more prediction errors can accumulate, making the prediction less reliable. As a result, less weight is given to the LSTM prediction $\hat{\lambda}^1_{c+k}$ and more reliance on the best-matched trajectory $\lambda^{b,r}_{c+k}$.

A potential issue with using $\alpha_{c+k}\hat{\lambda}^1_{c+k} + (1-\alpha_{c+k})\lambda^{b,r}_{c+k}$ is that when the predicted trajectory point is near the destination airport, the role of LSTM will diminish to zero. To still have LSTM play a role in prediction, we took an additional weighted sum of $\alpha_{c+k}\hat{\lambda}^1_{c+k} + (1-\alpha_{c+k})\lambda^{b,r}_{c+k}$ and the LSTM prediction $\hat{\lambda}^1_{c+k}$, weighted by ω and $1-\omega$. ω indicates the extent of using smoothing. $\omega = 0$ means no smoothing. $\omega = 1$ corresponds to full smoothing.

$$\alpha_{c+k} = \frac{d^{\text{GCD},c+k}_{\text{remain}}}{d^{\text{GCD},c}_{\text{remain}}} \tag{12}$$

$$\hat{\lambda}^2_{c+k} = \omega\left[\alpha_{c+k}\hat{\lambda}^1_{c+k} + (1-\alpha_{c+k})\lambda^{b,r}_{c+k}\right] + (1-\omega)\hat{\lambda}^1_{c+k} \tag{13}$$

$$\hat{\varphi}^2_{c+k} = \omega\left[\alpha_{c+k}\hat{\varphi}^1_{c+k} + (1-\alpha_{c+k})\varphi^{b,r}_{c+k}\right] + (1-\omega)\hat{\varphi}^1_{c+k} \tag{14}$$

The above prediction procedure iteratively generated future trajectory points, until we identified the first trajectory point that reached the boundary of the destination terminal airspace or fell within the boundary (Figure 8). Considering the random initialization of the weights of the LSTM neural network, a different model may be obtained each time we train the LSTM neural network using the same training data, which may result in a different trajectory. To account for the randomness, multiple LSTM models were trained for each historical trajectory. Thus, multiple predictions were performed, each corresponding to one trained LSTM model and generating one predicted trajectory.

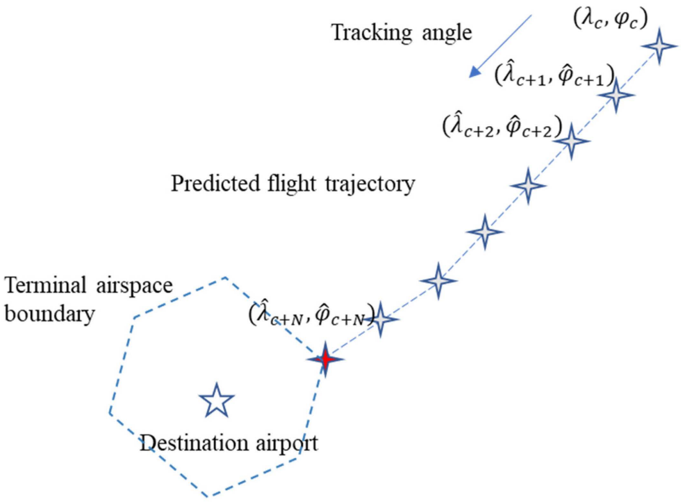

Tracking angle
(λ_c, φ_c)
$(\hat{\lambda}_{c+1}, \hat{\varphi}_{c+1})$
$(\hat{\lambda}_{c+2}, \hat{\varphi}_{c+2})$
Predicted flight trajectory
Terminal airspace boundary
$(\hat{\lambda}_{c+N}, \hat{\varphi}_{c+N})$
Destination airport

Figure 8. Illustration of when the iterative trajectory point prediction stops (when a trajectory point reaches the boundary of the destination terminal airspace or falls within the boundary).

As a final note, the way the LSTM-predicted trajectory points were adjusted is based on equal distance separation for trajectory points. As opposed to this, there would be a lack of basis for adjusting the LSTM-predicted trajectory points if they are required to be of equal time separation. This is because a predicted trajectory point is defined by longitude, latitude, and $d_{\text{remain}}^{\text{GCD}}$, but not time or speed information. Even if speed were part of the LSTM prediction, the predicted trajectory point location and speed may result in a time separation different from the prespecified time interval between the predicted and its preceding trajectory points (to our knowledge, there is no way to force the time separation computed based on the LSTM-predicted trajectory point location and speed to be equal to the prespecified time separation). Therefore, time separation inconsistency can occur. This, along with the way raw trajectory points were reconstructed, as explained in Section 2.3.1, justified the consideration of equal distance separation.

2.5. Ground Speed Prediction

GBM was used to predict flight ground speed at every trajectory point of the predicted remaining trajectory. GBM belongs to the class of decision-tree-based ensemble prediction methods that build a bucket of relatively simple models (also termed "base learners") to obtain a strong ensemble prediction. In particular, multiple single decision tree models were strategically combined rather than fitting the best single decision tree model. Compared with the random forest method, which is another popular decision-tree-based ensemble method based on bagging (rather than boosting), GBM adds new trees to the ensemble sequentially. At each iteration, GBM accounts for the error of the previously ensembled trees and tries to recover the error when performing prediction in the next tree. As such, the error keeps decreasing in the subsequent tree ensemble. In contrast, a random forest model trains multiple trees individually each with a subset of the training data. The final prediction comes from the average of all individual tree-based predictions. As a result of these differences, GBM has been reported to outperform the random forest method in prediction accuracy [15,35].

In this paper, multiple GBM models were trained using historical flight trajectories for each combination of flight number and aircraft type. To develop a GBM model, the speed at the reconstructed trajectory points was used, which was estimated in the following way: As shown in Figure 9, the first point on the reconstructed trajectory point sequence is the same as the first point on the raw trajectory point sequence. Thus, the speed of the first point on the reconstructed trajectory, v_g^0, is the same as the speed of the first point on the raw trajectory, $v_{g,raw}^0$. The speed at any intermediate point on the reconstructed trajectory is approximated via interpolation using the speeds of two neighboring raw trajectory points. For example, in Figure 9, $v_g^2 = v_{g,raw}^1 + \frac{|AB|}{a_2}\left(v_{g,raw}^2 - v_{g,raw}^1\right)$.

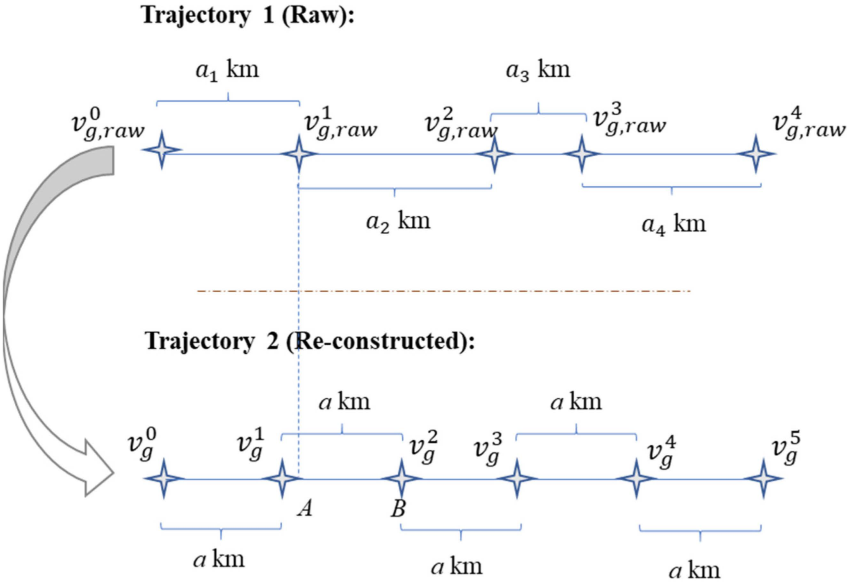


Figure 9. Estimation of speed at reconstructed trajectory points.

Using the notation $y_i = v_g^i$ and $\chi_i = \left(v_g^{i-1}, d_{\text{remain}}^{\text{GCD},i}\right)$ for trajectory point i in a reconstructed historical flight trajectory ($i = 2, \ldots, N$, where N denotes the total number of points in the trajectory), the prediction function to be trained is $y = F(\chi)$. GBM considers parameterizing $F(\chi)$ as follows:

$$F(\chi; \{\rho_m, a_m\}_1^M) = \sum_{m=1}^{M} \rho_m h(\chi; a_m) \tag{15}$$

where $\{\rho_m, a_m\}_1^M$ are the parameters of the GBM model. The values of ρ_m and a_m are iteratively obtained as shown later in Equations (18)–(20). M is the maximum number of iterations in performing the steepest descent to seek the best $F(\chi)$ that minimizes the L_2 square loss $L(y, F)$, which measures the deviation of the predicted value from the actual value of the output. The iterative relationship is as follows:

$$F_m(\chi) = F_{m-1}(\chi) - \rho_m \left\{ \left[\frac{\partial L(y, F(\chi_i))}{\partial F(\chi_i)} \right]_{F(\chi)=F_{m-1}(\chi)} \right\} \tag{16}$$

In Equation (15), $h(\chi; a_m)$, $m = 1, 2, \ldots, M$ are usually simple parameterized functions of χ, characterized by parameters a_m. Each $h(\chi; a_m)$ is called a "base learner", for which we specify the following regression tree:

$$h(\chi; a_m) = h\left(\chi; \left\{b_m^j, R_m^j\right\}_1^J\right) = \sum_{j=1}^{J} b_m^j 1\left(\chi \in R_m^j\right) \tag{17}$$

where $\{R_m^j\}_1^J$ are disjoint regions that collectively cover the space of all joint values of χ. These regions are represented by the terminal nodes of the corresponding tree. The indicator function $1(\cdot)$ takes value 1 if the argument is true, and 0 otherwise. b_m^j values are parameters of the base learner. In each iteration, a new tree is added to $F(\chi)$.

The question in the mth iteration is to identify a_m such that $h(\chi; a_m)$ is most parallel to (most correlated with) $-\left\{ \left[\frac{\partial L(y, F(\chi_i))}{\partial F(\chi_i)} \right]_{F(\chi)=F_{m-1}(\chi)} \right\}$. This can be obtained from solving the following least-square minimization problem, which is well studied with standard procedures.

$$a_m = \underset{a,\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[- \left[\frac{\partial L(y, F(\chi_i))}{\partial F(\chi_i)} \right]_{F(\chi)=F_{m-1}(\chi)} - \beta h(\chi_i; a) \right]^2 \tag{18}$$

The resulting $h(\chi_i; a_m)$ is used to replace $-\left[\frac{\partial L(y,F(\chi_i))}{\partial F(\chi_i)}\right]_{F(\chi)=F_{m-1}(\chi)}$ in the steepest descent expression in Equation (16). A new line search is conducted to determine the value for ρ_m:

$$\rho_m = \underset{\rho}{\operatorname{argmin}} L(y, F_{m-1}(\chi) + \rho h(\chi; a_m)) \tag{19}$$

which is then used to update $F(\chi)$:

$$F_m(\chi) = F_{m-1}(\chi) + v\rho_m h(\chi; a_m) \tag{20}$$

where $v \in (0, 1]$ is the learning rate. The process using Equations (18)–(20) is called "boosting". Because gradient is used in the steepest descent, the overall procedure is named "gradient boosting".

To summarize, the GBM algorithm can be represented as follows in Algorithm 1:

Algorithm1: GBM Algorithm.

```
Initialization : $F_0(\chi) = \operatorname{argmin}_{\rho} \sum_{i=1}^{N} L(y_i, \rho)$
For $m = 1$ to $M$ do:
    $a_m = \underset{a,\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ - \left[ \frac{\partial L(y,F(\chi_i))}{\partial F(\chi_i)} \right]_{F(\chi)=F_{m-1}(\chi)} - \beta h(\chi_i; a) \right]^2$
    $\rho_m = \underset{\rho}{\operatorname{argmin}} L(y, F_{m-1}(\chi) + \rho h(\chi; a_m))$
    $F_m(\chi) = F_{m-1}(\chi) + v\rho_m h(\chi; a_m)$
End for
```

2.6. ETA_TAB and ELDT Prediction

After the prediction of flight trajectory and aircraft ground speed, discussed in Sections 2.4 and 2.5, the airborne time for the remaining trajectory was computed. Specifically, let (λ_c, φ_c) and v_g^c denote the current position and ground speed of an aircraft (shown in Figure 10). $(\lambda_{c+1}, \varphi_{c+1}), \cdots, (\lambda_{c+N}, \varphi_{c+N})$ and $v_g^{c+1}, \cdots, v_g^{c+N}$ denote the predicted trajectory points and ground speeds, where N is the number of points before reaching the boundary of the destination terminal airspace. Between two neighboring trajectory points, flight time is calculated by dividing the distance of the segment (which is $a = 1$ nm) by the average ground speed, which is approximated as the average of the predicted ground speeds at the two points (Equation (21)). The total remaining airborne time up to the terminal airspace boundary, $\hat{T}_1$, is the sum of flight times between each of two neighboring trajectory points up to the last point N. ETA_TAB is the current time t_0 plus $\hat{T}_1$. These are shown in Equations (22) and (23).

$$\overset{\_\,i}{v}_g = \frac{v_g^{i-1} + v_g^{i}}{2} \tag{21}$$

$$\hat{T}_1 = \sum_{i}^{N} \frac{a}{\overset{\_\,i}{v}_g} \tag{22}$$

$$ETA\_TAB = t_0 + \hat{T}_1 \tag{23}$$

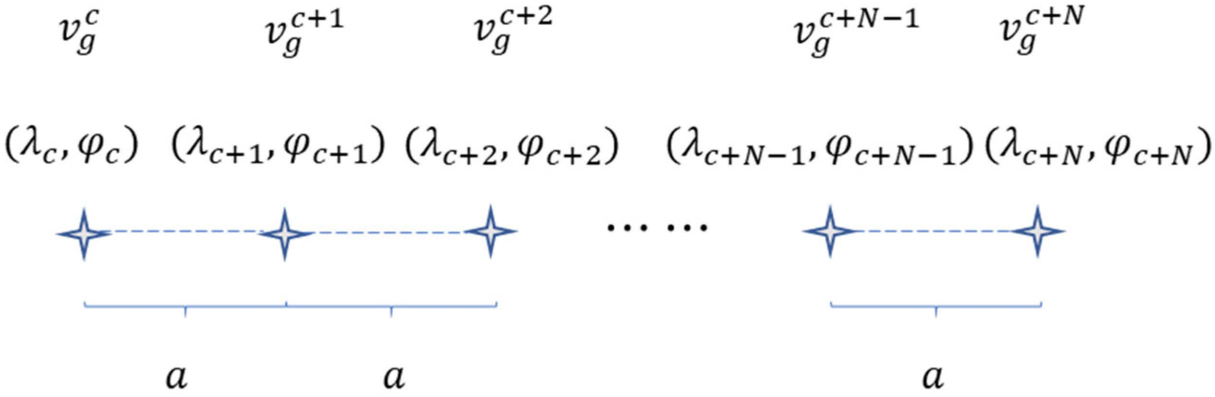

v_g^c
v_g^{c+1}
v_g^{c+2}
v_g^{c+N-1}
v_g^{c+N}
(λ_c, φ_c)
$(\lambda_{c+1}, \varphi_{c+1})$
$(\lambda_{c+2}, \varphi_{c+2})$
$(\lambda_{c+N-1}, \varphi_{c+N-1})$
$(\lambda_{c+N}, \varphi_{c+N})$
…… …
a
a
a

Figure 10. Trajectory and ground speed prediction outputs for remaining airborne time.

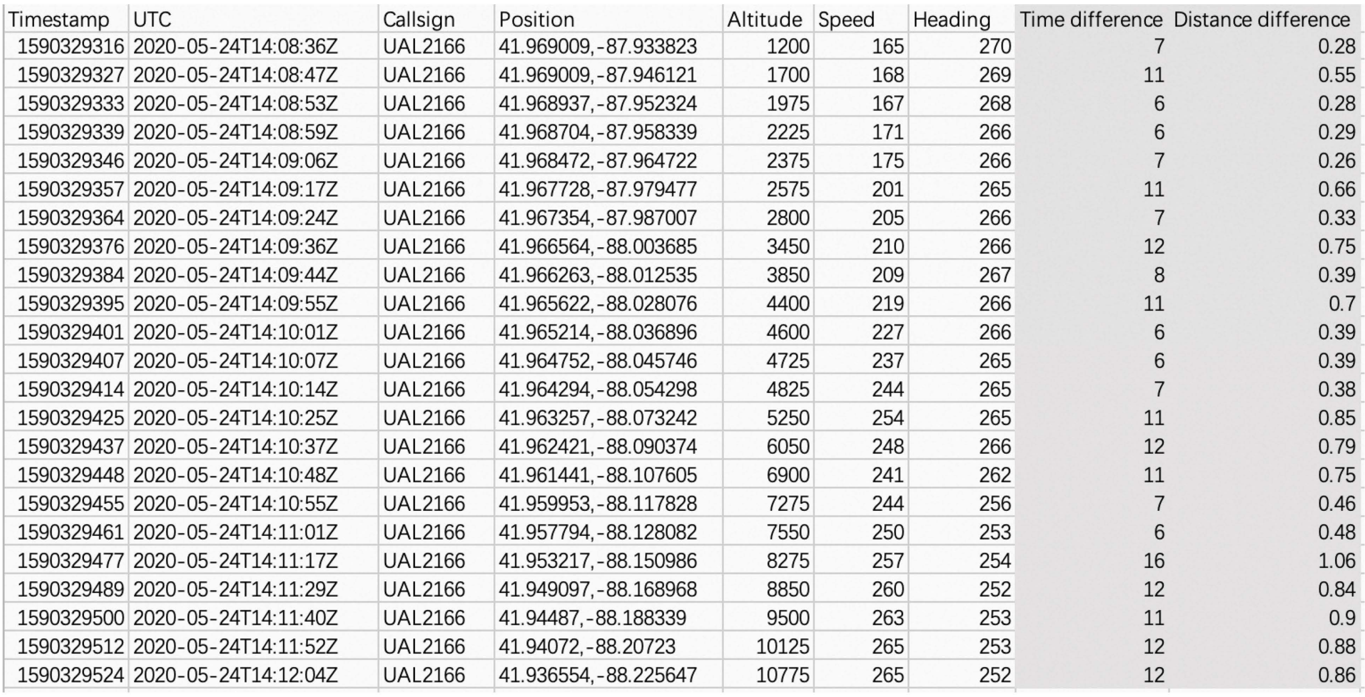
When entering the terminal airspace, a flight's trajectory will be subject to greater variability [36]. Air traffic controllers may assign different procedures and maneuvers to different flights making it more challenging to reliably predict flight trajectory within the terminal airspace. In addition, within the terminal airspace, flight ground speed decreases drastically. To account for these uncertainties, we opted for the random sampling of a historical terminal approach time $\hat{T}_2$ among all historical trajectories of the same OD, aircraft type, and approach entry position (a terminal airspace typically has a few fixed approach entry positions (this is confirmed by referring to the raw flight trajectories we present in Section 3). If a predicted trajectory did not exactly touch any approach entry position, then we considered the approach entry position that was the closest to the touch point of the predicted trajectory to the terminal airspace boundary, as the approach entry position of the predicted trajectory) and add the sampled time to ETA_TAB to obtain ELDT.

$$ELDT = ETA\_TAB + \hat{T}_2 \tag{24}$$

3. Demonstration of the Prediction Approach

3.1. Data

To demonstrate the proposed approach, in this paper, we used flights of two airport pairs for ETA_TAB and ELDT prediction: Denver to San Francisco (DEN-SFO) and Chicago O'Hare to San Francisco (ORD-SFO), which have GCDs of 839 nm and 1600 nm, respectively. The involved airports are among the busiest airports in the US. Flight trajectory data were collected from Flightradar24 for the period between January 2020 and June 2020. In total, 63 and 136 flight records were collected from DEN-SFO and ORD-SFO, for flight numbers UA1497 and UA2166, respectively. Each flight record was composed of a sequence of trajectory points. Each trajectory point contained information on seven items: monitoring time in Unix timestamp, monitoring time in coordinated universal time (UTC), flight callsign (i.e., flight number), position information (longitude and latitude), altitude, ground speed, and heading. An excerpt of the flight records is shown in Figure 11. For a point, its longitude and latitude together with the longitude and latitude of the destination airport (collected from OpenFlights) were used to calculate $d_{\text{remain}}^{\text{GCD}}$. Because SFO was the destination airport, information about the SFO terminal airspace boundary (i.e., TRACON boundary) was collected from the official Federal Register documentation [37].



Timestamp	UTC	Callsign	Position	Altitude	Speed	Heading	Time difference	Distance difference
1590329316	2020-05-24T14:08:36Z	UAL2166	41.969009,-87.933823	1200	165	270	7	0.28
1590329327	2020-05-24T14:08:47Z	UAL2166	41.969009,-87.946121	1700	168	269	11	0.55
1590329333	2020-05-24T14:08:53Z	UAL2166	41.968937,-87.952324	1975	167	268	6	0.28
1590329339	2020-05-24T14:08:59Z	UAL2166	41.968704,-87.958339	2225	171	266	6	0.29
1590329346	2020-05-24T14:09:06Z	UAL2166	41.968472,-87.964722	2375	175	266	7	0.26
1590329357	2020-05-24T14:09:17Z	UAL2166	41.967728,-87.979477	2575	201	265	11	0.66
1590329364	2020-05-24T14:09:24Z	UAL2166	41.967354,-87.987007	2800	205	266	7	0.33
1590329376	2020-05-24T14:09:36Z	UAL2166	41.966564,-88.003685	3450	210	266	12	0.75
1590329384	2020-05-24T14:09:44Z	UAL2166	41.966263,-88.012535	3850	209	267	8	0.39
1590329395	2020-05-24T14:09:55Z	UAL2166	41.965622,-88.028076	4400	219	266	11	0.7
1590329401	2020-05-24T14:10:01Z	UAL2166	41.965214,-88.036896	4600	227	266	6	0.39
1590329407	2020-05-24T14:10:07Z	UAL2166	41.964752,-88.045746	4725	237	265	6	0.39
1590329414	2020-05-24T14:10:14Z	UAL2166	41.964294,-88.054298	4825	244	265	7	0.38
1590329425	2020-05-24T14:10:25Z	UAL2166	41.963257,-88.073242	5250	254	265	11	0.85
1590329437	2020-05-24T14:10:37Z	UAL2166	41.962421,-88.090374	6050	248	266	12	0.79
1590329448	2020-05-24T14:10:48Z	UAL2166	41.961441,-88.107605	6900	241	262	11	0.75
1590329455	2020-05-24T14:10:55Z	UAL2166	41.959953,-88.117828	7275	244	256	7	0.46
1590329461	2020-05-24T14:11:01Z	UAL2166	41.957794,-88.128082	7550	250	253	6	0.48
1590329477	2020-05-24T14:11:17Z	UAL2166	41.953217,-88.150986	8275	257	254	16	1.06
1590329489	2020-05-24T14:11:29Z	UAL2166	41.949097,-88.168968	8850	260	252	12	0.84
1590329500	2020-05-24T14:11:40Z	UAL2166	41.94487,-88.188339	9500	263	253	11	0.9
1590329512	2020-05-24T14:11:52Z	UAL2166	41.94072,-88.20723	10125	265	253	12	0.88
1590329524	2020-05-24T14:12:04Z	UAL2166	41.936554,-88.225647	10775	265	252	12	0.86

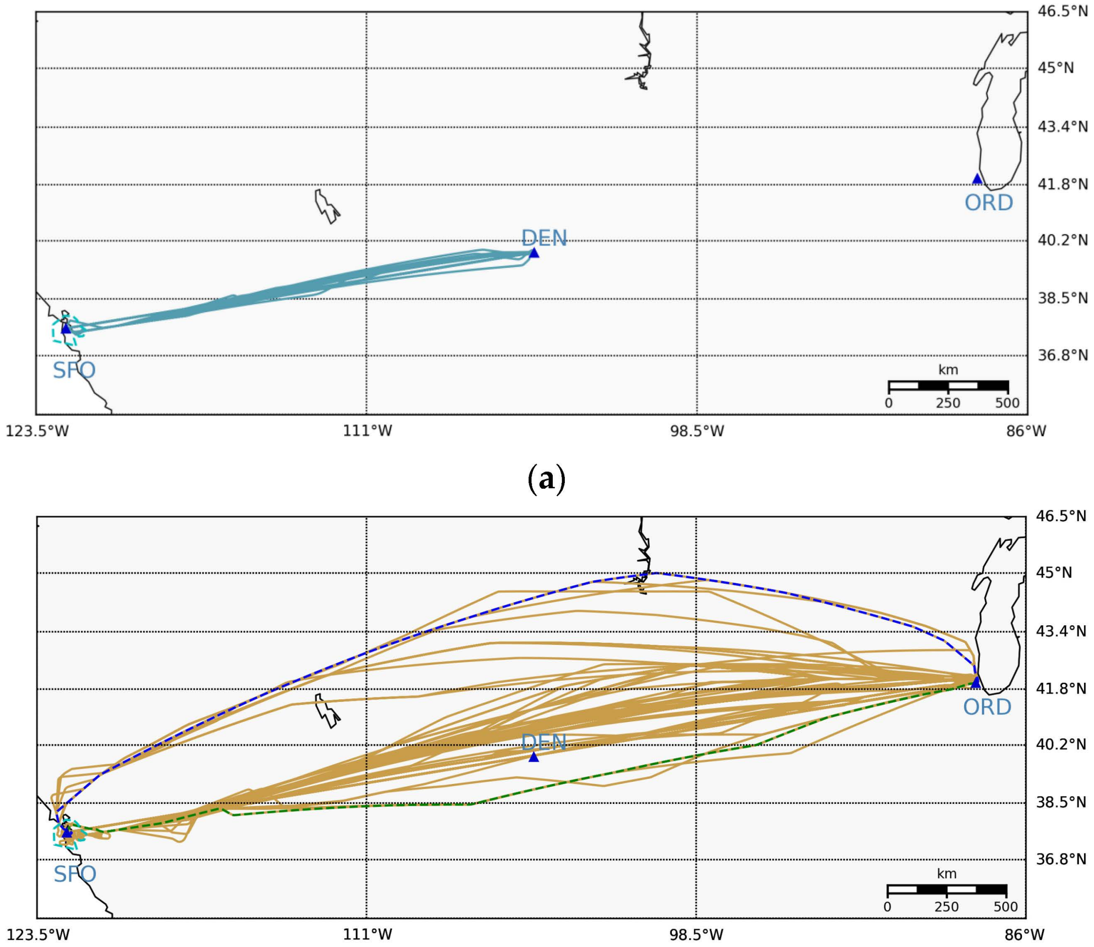
Figure 11. Sample flight trajectory records in our data.

Building on the raw data, we further added two columns on the right (shown in grey in Figure 11) which indicate (1) the time difference (in second) and distance difference (in nm) between a trajectory point and its immediately preceding trajectory point (the row immediately above the current row). These differences show that the trajectory points were unevenly spaced in both time and distance.

In the rest of this section, we first present the ETA_TAB and ELDT prediction results for the two flights in detail: UA1497 on 20 April 2020 from DEN-SFO (flown by a B777-200 aircraft), denoted as FLT1, and UA2166 on 9 May 2020 from ORD-SFO (flown by an A320 aircraft), denoted as FLT2. To further investigate the validity of the proposed approach, we also report the results for 30 randomly picked flights not used in training from each airport pair. ETA_TAB and ELDT predictions were performed at three airborne positions: when a flight was at 25th, 50th, and 75th GCD percentiles. In other words, we predicted ETA_TAB and ELDT with the assumption that a flight has flown 25%, 50%, and 75% of the GCD. The prediction was performed on a laptop with 16 GB RAM, Intel Core i7-9850H (2.60 GHz), and NVIDIA Quadro T2000 graphics running in a Windows environment.

3.2. Trajectory Matching

Figure 12a,b show the 63 and 136 flight trajectories of the two airport pairs. For DEN-SFO, it can be seen that all the trajectories are quite similar. In contrast, ORD-SFO flight trajectories exhibit more differences, suggesting the importance of identifying the best-matched trajectory. For example, in Figure 12b, it would be unlikely to yield an accurate prediction if using the purple dashed trajectory in the north to train an LSTM model and then using the trained model to predict the trajectory shown in the dashed green in the south. According to the FAA, the width of an air route extends 4 nm on each side from the centerline [38]. Thus, we set $\varepsilon = 8$ nm as the threshold for determining whether two trajectory points were close enough in trajectory matching.



(**a**)

(**b**)

Figure 12. Flight trajectory records for the median- and long-haul airport pairs: (**a**) flight trajectory records of UA1497 from DEN-SFO from 1 January to 30 June 2020; (**b**) flight trajectory records of UA2166 from ORD-SFO from 1 January to 30 June 2020.

For each of FLT1 and FLT2, we first reconstructed the trajectory point sequence so that trajectory points were of equal distance. Because the prediction was performed from the current position, the best-matched historical trajectory depended on what had been flown by the flight under study. For example, in the beginning, a flight's trajectory may be most similar to the trajectory of a historical flight A. But as the flight continues flying, its trajectory may become most similar to the trajectory of a different historical flight B. Table 1 presents the trajectory matching results for FLT1 and FLT2, when the flights were at 25th, 50th, and 75th GCD percentiles. We found that the best-matched trajectory for FLT1 was always the trajectory of UA1497 on 19 April 2020 (with a similarity score of one). For FLT2, which flew longer, the best-matched trajectory was the trajectory of UA2166 on 13 February 2020 when the flight was at the 25th GCD percentile. The best-matched trajectory changed to that of UA2166 on 19 January 2020 at the 50th and 75th GCD percentiles. The similarity score was reduced from 1 to 0.99. Figures 13 and 14 illustrate the best-matched trajectories (indicated by the blue dashed lines) at the three GCD percentiles for FLT1 and FLT2.

Table 1. Trajectory matching results of FLT1 and FLT2.

Airport Pair	Current Flight	Current Position	Best-Matched Historical Flight	Similarity Score	Matching Time (min)
DEN-SFO	FLT1 (UA1497(4/20/20))	25th GCD percentile	UA1497 (4/19/20)	1.00	0.01
		50th GCD percentile	UA1497 (4/19/20)	1.00	0.04
		75th GCD percentile	UA1497 (4/19/20)	1.00	0.09
ORD-SFO	FLT2 (UA2166(5/9/20))	25th GCD percentile	UA2166 (3/3/20)	1.00	0.28
		50th GCD percentile	UA2166 (1/19/20)	1.00	0.66
		75th GCD percentile	UA2166 (1/19/20)	0.99	1.05

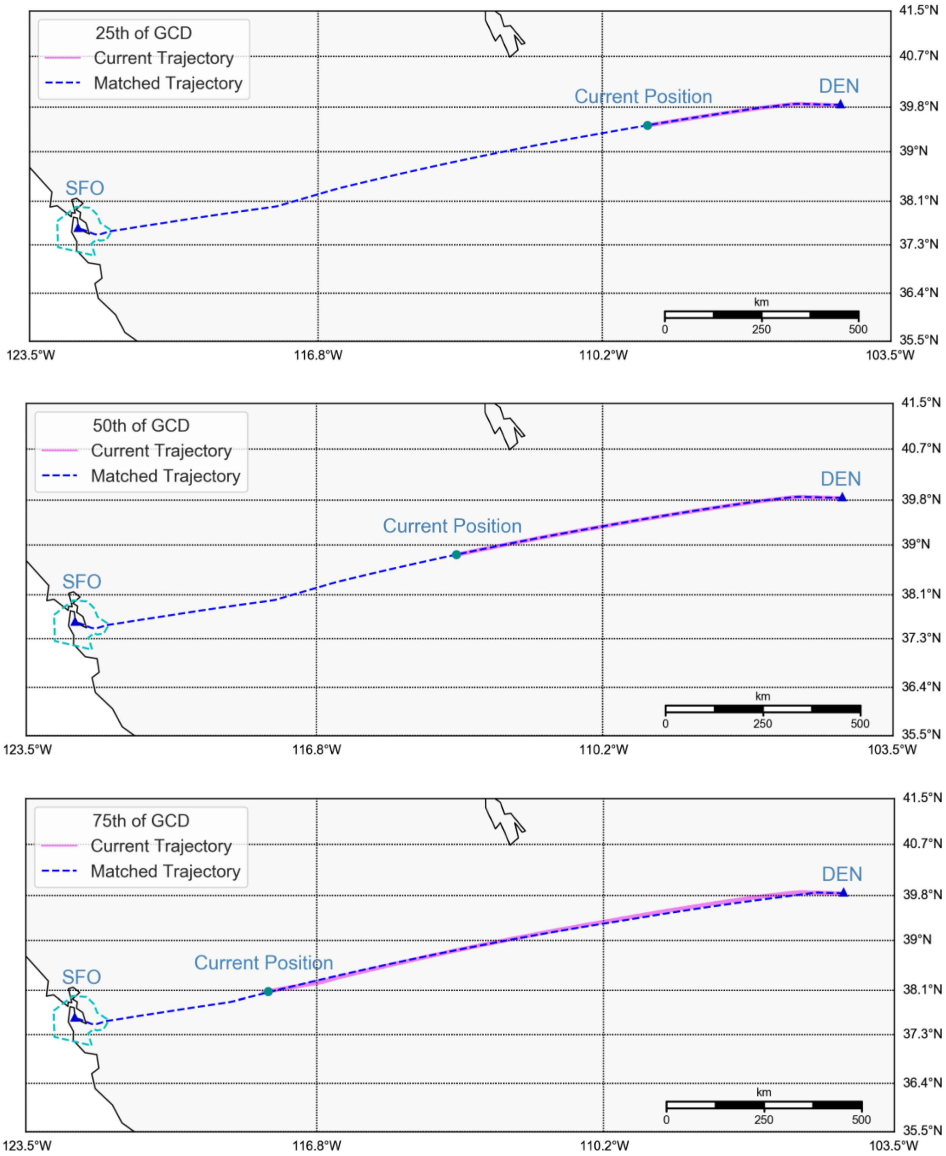


Figure 13. The current flight's trajectory (in solid pink) with the best-matched historical flight trajectory (in dashed blue) for FLT1.

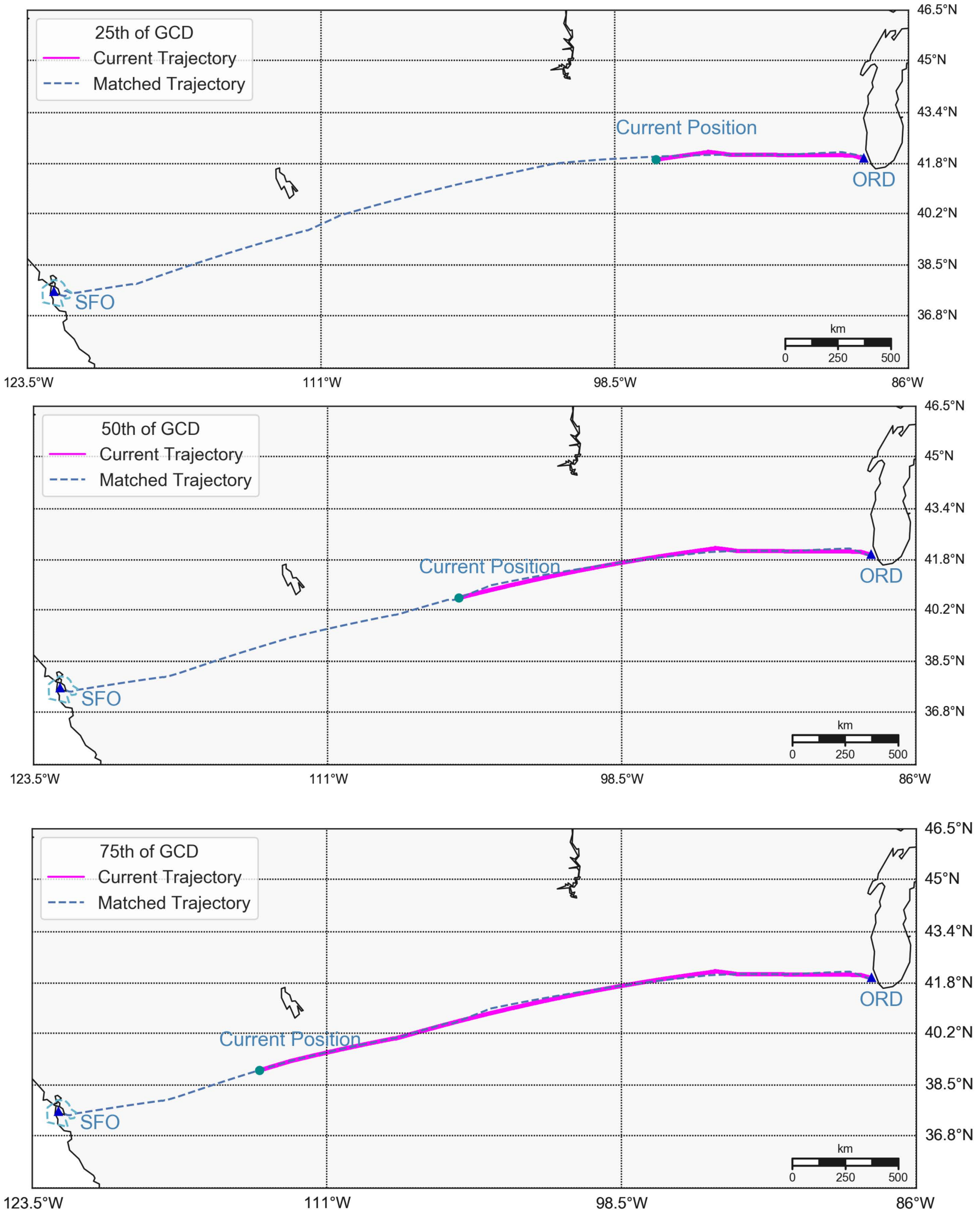


Figure 14. The current flight's trajectory (in solid pink) with the best-matched historical flight trajectory (in dashed blue) for FLT2.

In terms of the time needed for trajectory matching, it was quite small, as reported in the last column of Table 1. For comparison, we also experimented with using dynamic time warping (DTW) to identify the best-matched trajectories. It was found that DTW yielded the same best-matched trajectories for FLT1 and FLT2. On the other hand, the computation time was longer. For example, the best trajectory matching for FLT2 using DTW took 3.67 min, 7.79 min, and 12.79 min when the flight had flown 25%, 50%, and 75% of GCD. On the other hand, if a much larger number of historical trajectories were considered for matching, matching time might increase. In this case, similar historical flight trajectories could be clustered before matching. A representative trajectory from the best-matched cluster could then be picked for LSTM training. This process could help reduce the time for matching.

3.3. Trajectory Prediction

For each airport pair, we used flight trajectories that occurred prior to FLT1 (FLT2) to train the respective LSTM models. As mentioned in Section 2.4.2, to account for the random initialization of the weights of the LSTM neural network, multiple LSTM models were trained for each airport pair. Each trained model was used to generate one predicted trajectory. We set the number of LSTM models to be trained for an airport pair to 30. LSTM model training was performed using Keras with a TensorFlow backend [39]. An Adam optimizer with adaptive learning rates was used to update the network weights iteratively [40].

Prior to training the LSTM models, the hyperparameters, including the learning rate and the number of iterations, were tuned with grid search. For each airport pair, the historical trajectories were split, with 80% as training data and the remaining 20% as validation data. Given that only historical trajectories were used, the training process can be carried out offline. The training time of the 30 LSTM models for DEN-SFO was 50.2 min, and of the 30 LSTM models for ORD-SFO was 568.8 min. To perform a comparison, standard RNN and gated recurrent unit (GRU) models were also trained in a similar fashion and used for prediction. The RMSE of training and validation results are presented in Table 2. Overall, LSTM was found to perform the best among the three models.

Table 2. RMSE of LSTM, RNN, and GRU models for trajectory prediction.

Methods	**Features**	**Training Data**		**Validation Data**	
		DEN-SFO	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**
LSTM	Longitude (degree)	1.04×10^{-4}	3.22×10^{-4}	2.52×10^{-4}	4.09×10^{-4}
	Latitude (degree)	2.60×10^{-5}	6.86×10^{-5}	1.36×10^{-5}	4.43×10^{-4}
	$d_{\text{remain}}^{\text{GCD}}$ (nm)	0.33	0.84	0.10	1.37
RNN	Longitude (degree)	4.18×10^{-3}	5.11×10^{-3}	1.42×10^{-3}	7.06×10^{-3}
	Latitude (degree)	6.31×10^{-3}	4.10×10^{-3}	3.06×10^{-3}	3.93×10^{-3}
	$d_{\text{remain}}^{\text{GCD}}$ (nm)	1.71	4.29	1.30	3.35
GRU	Longitude (degree)	2.87×10^{-4}	7.68×10^{-4}	2.93×10^{-4}	8.14×10^{-4}
	Latitude (degree)	5.29×10^{-5}	3.46×10^{-4}	2.12×10^{-4}	7.09×10^{-4}
	$d_{\text{remain}}^{\text{GCD}}$ (nm)	0.85	1.59	0.79	1.41

With the trained LSTM models, the procedure described in Section 2.4.2 was followed for trajectory prediction in which smoothing is one of the key techniques. Recall that in doing so, ω is a parameter indicating the extent of using smoothing. When ω is close to 0, the predicted trajectories cannot enter the terminal airspace in some cases. After smoothing was performed, the predicted trajectories could enter the terminal airspace in a reasonable position. To test the sensitivity of the trajectory prediction to ω, we varied the value of ω from 0.2 to 0.8 in an increment of 0.1. For each OD, we randomly picked 20 flights for trajectory prediction. We found that, as ω increased from 0.2 to 0.5, the percentage of predicted trajectories entering the terminal airspace increased from 93.3% to 100% (recall that for a given flight, 30 LSTM models were trained, thus generating 30 predicted trajectories, so in total, 600 predicted trajectories were generated). Once ω exceeded 0.5, 100% of the predicted trajectories always entered the terminal airspace. We further measured the average distance between the predicted trajectory point and the corresponding actual trajectory point (i.e., the point that had the same position number in their respective trajectory point sequences). We found that as ω increased from 0.2 to 0.5, this average distance decreased, suggesting improvement in the prediction accuracy. Once ω exceeded 0.5, the average distance stayed more or less the same. In view of these results, and to have the LSTM-predicted trajectory play an important role, we set $\omega = 0.5$.

The time needed for trajectory matching and trajectory prediction was 0.20, 0.24, and 0.27 min at the 25th, the 50th, and the 75th GCD percentiles for FLT1; and 1.32, 1.50, and 1.66 min at the 25th, the 50th, and the 75th GCD percentiles for FLT2. Figures 15 and 16 present the predicted flight trajectories of FLT1 and FLT2. In each figure, a plot corresponds to the prediction assuming the corresponding flight is at the 25th, 50th, and 75th GCD-percentile positions. The orange curves denote the actual flight trajectories (ground truth). The blue dashed curves represent the predicted flight trajectories that can reach the terminal airspace boundary. We found that the predicted trajectories coincided quite well with the actual trajectories.

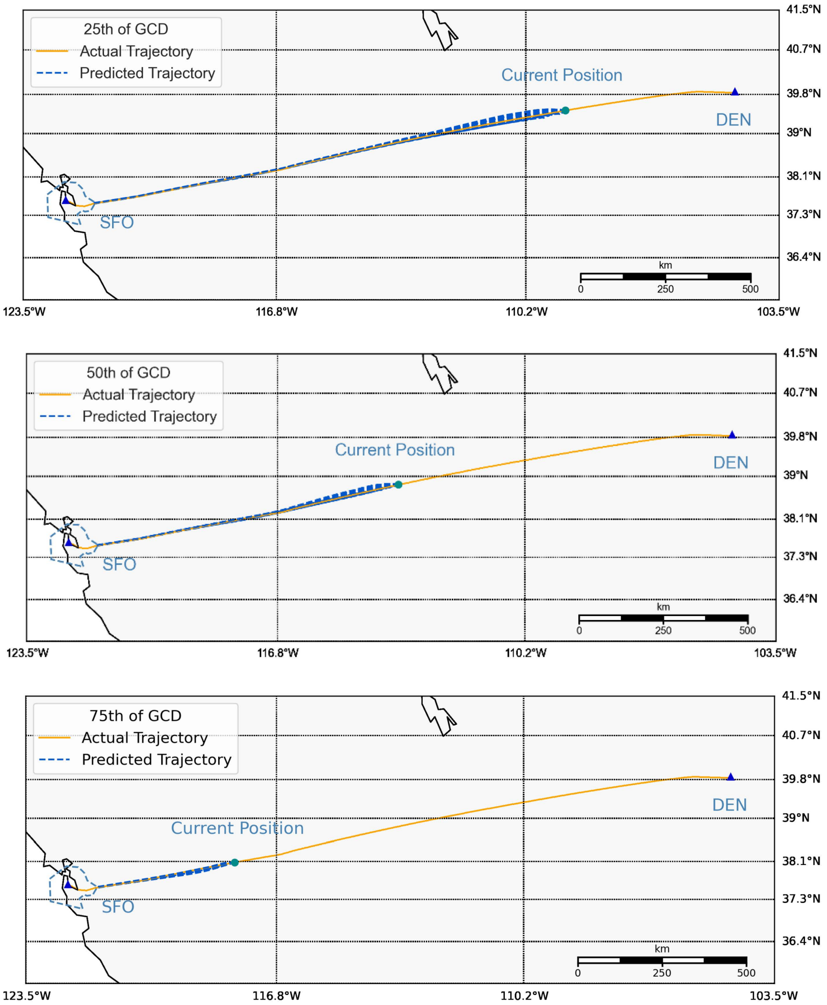


Figure 15. The predicted trajectories for FLT1 at 25th, 50th, and 75th GCD percentiles.

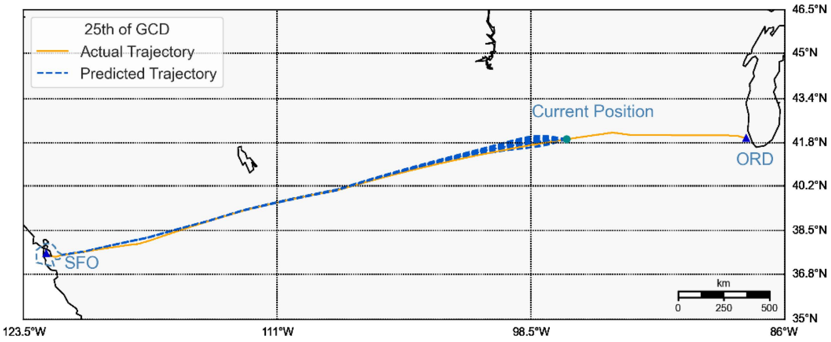


Figure 16. *Cont.*

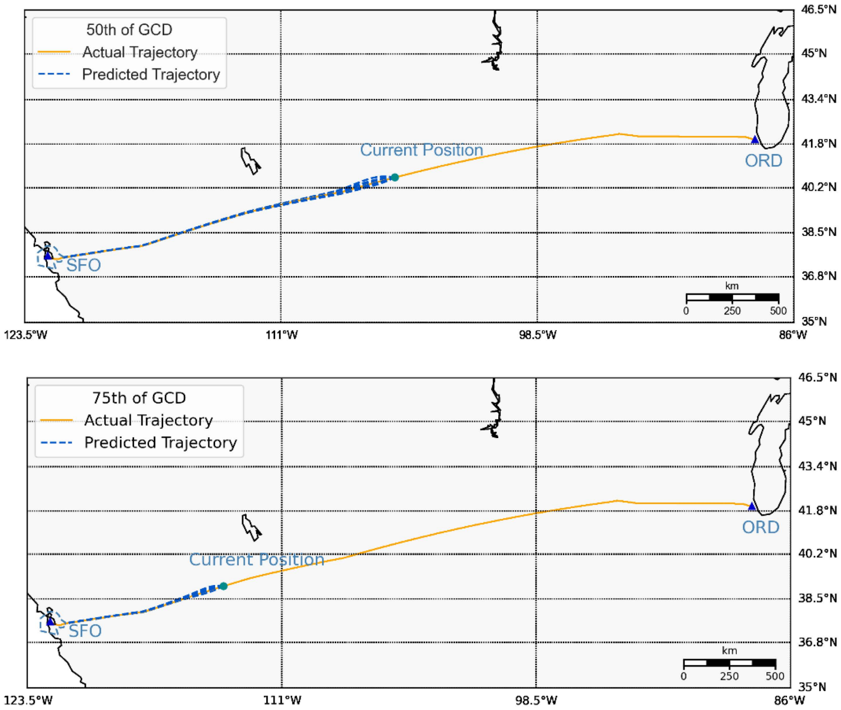


Figure 16. The predicted trajectory for FLT2 at 25th, 50th, and 75th GCD percentiles.

To further examine the quality of trajectory prediction, the distance between each predicted trajectory point and the corresponding actual trajectory point (i.e., the point that had the same order position as the predicted point in the actual trajectory point sequence) was calculated. The distributions of these distances are shown in Figure 17 (left for FLT1 and right for FLT2). Note that each distribution plot includes the trajectory points on all the predicted trajectories in Figures 15 and 16. The distributions for both flights are right-skewed, with the average being 3.17, 2.81, and 1.50 nm for FLT1; and 3.97, 1.11, and 0.95 nm for FLT2, when predicting at 25th, 50th, and 75th GCD percentiles. The distributions and their small averages relative to flight ground speed during the cruise (around 432 nm/h; see Appendix A, Figure A1) suggest reasonably accurate trajectory predictions using the LSTM models.

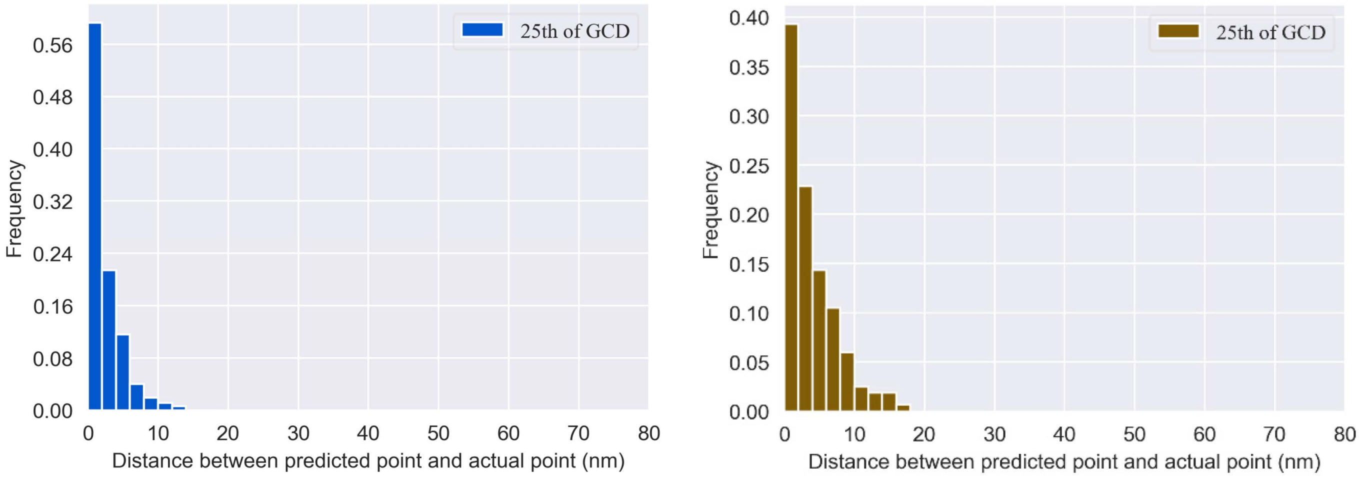
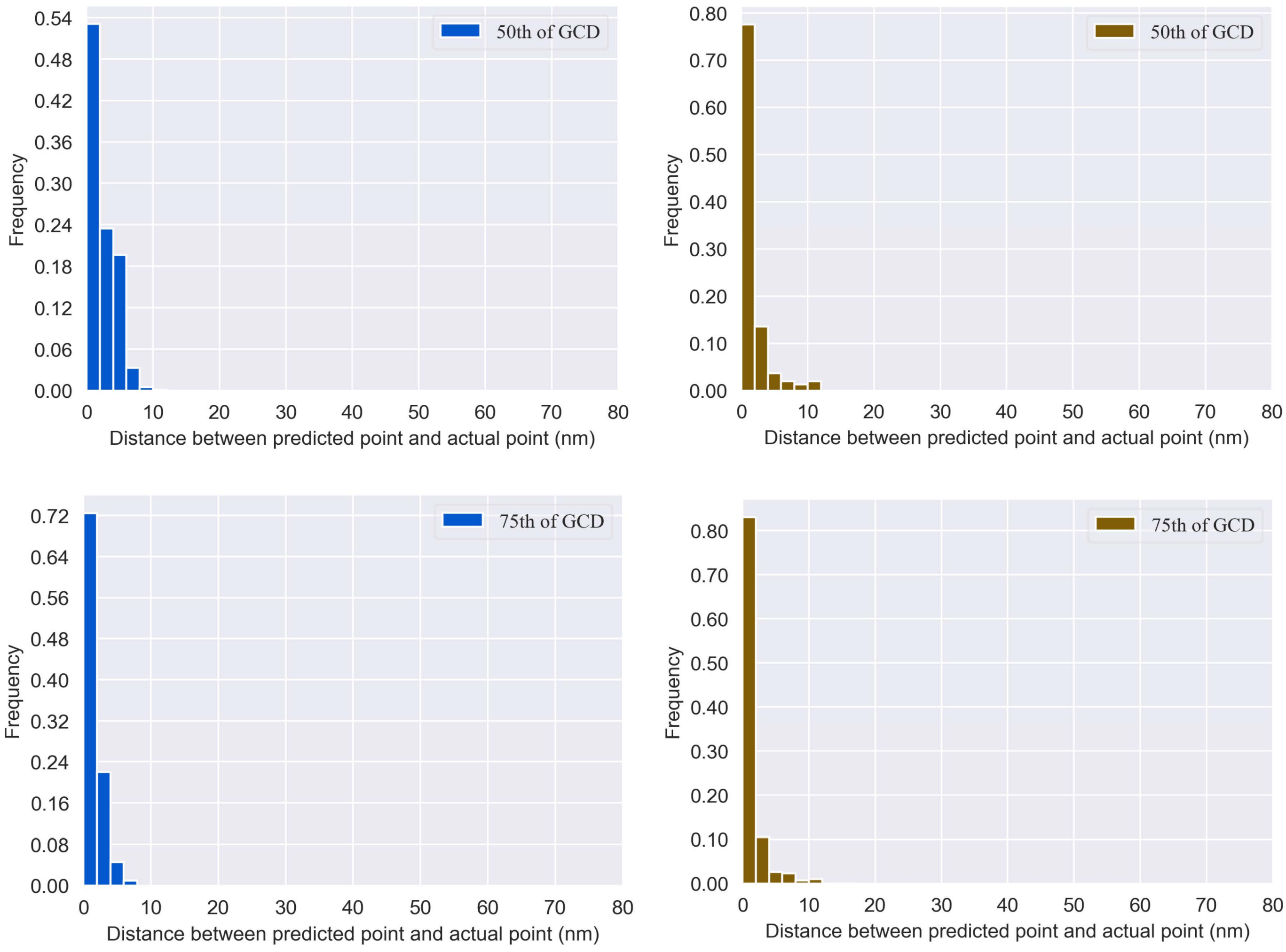


Figure 17. *Cont.*



FLT1 FLT2

Figure 17. Distributions of the distance between the predicted and actual trajectory points (the two points have the same order position in their respective trajectory point sequences) for FLT1 (left) and FLT2 (right), predicting at 25th, 50th, and 75th GCD percentiles.

3.4. Ground Speed Prediction

Following the description in Section 2.5, GBM models were trained using reconstructed trajectory points of all the historical trajectories of the same flight number and the same aircraft type. Given that each airport pair in our numerical experiments had only one flight number, an airport pair was associated with one trained GBM model. In GBM training, a data record included a reconstructed trajectory point i (except for the first point of a trajectory) consisting of ground speed at the previous trajectory point (v_g^{i-1}), the remaining GCD of the current point ($d_{\text{remain}}^{\text{GCD},i}$), and ground speed at the current trajectory point (v_g^i).

Four hyperparameters need to be tuned in GBM training, the number of regression trees (M), the maximum depth of a tree, the minimum sample leaf of a tree (i.e., the minimum number of observations a node needs to have to be considered for splitting), and the learning rate (v) were used. The following procedure was adopted in training: For an airport pair, the historical trajectory data were divided into five sets of equal size (i.e., five folds). Considering that each hyperparameter could take several possible values, we used a grid search of possible combinations of hyperparameter values. For a given combination, we drew four folds from the data to train a GBM model. Then, the trained model was applied to the remaining fold of the data to compute RMSE. Since we had five folds, there were five possible ways to draw four folds of the data, resulting in five GBM models and five RMSEs. The five RMSEs were averaged. We repeated this for all combinations of hyperparameter values and chose the combination that yielded the smallest average RMSE. The corresponding GBM model was selected for ground speed prediction.

Similar to LSTM model training, the training of GBM models could be performed offline given that only historical trajectories were used. For FLT1 (on 20 April 2020) and FLT2 (on 9 May 2020), 17 flights with B777 series before 4/20/2020 for DEN-SFO and 6 flights with A320 series before 5/9/2020 for ORD-SFO were used for training. The training took 11.03 min for DEN-SFO and 47.84 min for ORD-SFO per aircraft type on average. Once the GBM model for an airport pair was selected, it was applied to all the historical trajectory data that were used for training, to calculate the RMSE for the predicted ground speed. The resulting RMSEs were 2.2 nm/h for DEN-SFO and 2.3 nm/h for ORD-SFO.

The trained GBM models were further applied, in an iterative manner, to the predicted remaining trajectory of FLT1 and FLT2. More specifically, standing at the current trajectory point, we predicted the ground speed of the first predicted trajectory point using ground speed at the current point and $d_{\text{remain}}^{\text{GCD}}$ of the first predicted trajectory point. Then, we used the predicted ground speed of the first predicted trajectory point and $d_{\text{remain}}^{\text{GCD}}$ of the second predicted trajectory point, to predict the ground speed of the second predicted trajectory point. We calculated RMSEs with respect to ground speeds along the reconstructed trajectory point sequence of the true trajectory. (It may be possible that the number of reconstructed trajectory points on the true trajectory would be different from the number of trajectory points on the predicted trajectory. If this occurred, we considered the smaller number of the two while calculating RMSE.) Since multiple predicted trajectories were generated for each flight, and each predicted trajectory yielded one RMSE, the RMSEs were averaged. The averaged RMSEs when predicting at 25th, 50th, and 75th GCD percentiles are shown in Table 3. Because ORD-SFO had a larger GCD than DEN-SFO, greater prediction uncertainties were expected. Thus, it was not surprising that the average RMSE for FLT2 was larger than for FLT1. To make a comparison, random forest models were also trained in the same fashion as RMSEs reported in Table 3. It can be seen that the GBM performed better than the random forest model.

Table 3. Averaged RMSE (in nm/h) overall multiple trajectory predictions of the predicted ground speed for points on the predicted remaining trajectory for FLT1 and FLT 2 at different GCD percentiles.

	GBM		**Random Forest**	
	FLT1	**FLT2**	**FLT1**	**FLT2**
25th GCD percentile	7.36	10.01	7.63	12.04
50th GCD percentile	9.41	12.98	9.42	13.97
75th GCD percentile	9.83	13.56	10.4	15.29

It is worth noting that the RMSEs in Table 3 are larger than the ones mentioned in two paragraphs above (2.2 nm/h for DEN-SFO and 2.3 nm/h for ORD-SFO), which is not surprising as here, each speed prediction is based on a predicted trajectory point and predicted speed of the preceding trajectory point, which was also predicted (except for the first prediction, which was performed at the current trajectory point). Thus, prediction errors could propagate along the predicted trajectory. By contrast, the RMSEs in the two paragraphs above are based on ground truth information for both the current trajectory point and the speed of the preceding trajectory point. Despite this, the RMSEs remained relatively small considering the magnitude of flight speed during the cruise (around 432 nm/h, as shown in Appendix A). We also observed a slightly increasing trend of RMSE with greater GCD percentiles. A possible explanation is that near the end of the predicted trajectory (approaching the terminal airspace), a flight starts to descend, resulting in a less accurate prediction of ground speed. With a smaller GCD percentile, the portion of such predictions is smaller, leading to a smaller RMSE.

3.5. ETA_TAB and ELDT Prediction

3.5.1. ETA_TAB Prediction

As multiple predicted trajectories were generated for each of FLT1 and FLT2 at a GCD percentile, multiple ETA_TABs were obtained following Equations (21)–(23). We took the median of these ETA_TABs as the point estimate. We chose median over mean as the median is more robust to outliers in multiple predictions. We compared the ETA_TABs from our approach with two baseline estimates. The first baseline estimate (referred to as "best match") was obtained by adding the airborne time to the destination terminal airspace boundary of the best-matched historical flight to the actual wheels-off time. For the second baseline estimate (referred to as "average"), the average airborne time from wheels-off to arriving at the destination terminal airspace boundary was used for all historical flights of the same OD, flight number, and aircraft type, and then the average time to the actual wheels-off time of the current flight was added as ETA_TAB. We calculated the difference between each ETA_TAB and the actual arrival time at the terminal airspace boundary, termed Δt_{term} (ETA_TAB minus the actual arrival time).

Table 4 presents Δt_{term} for FLT1 and FLT2 at the three GCD percentiles. We found that the proposed approach yielded very accurate predictions—indeed significantly better than the two baseline estimates. Overall, the absolute difference between the estimated ETA_TAB and the actual arrival time was always below one minute (with only one exception of ORD-SFO at the 50th GCD percentile).

Table 4. Δt_{term} of FLT1 and FLT2 using the proposed approach and two baseline estimates, at the three GCD percentiles.

	25th GCD Percentile		**50th GCD Percentile**		**75th GCD Percentile**	
	DEN-SFO	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**
Proposed approach	−0.43	−0.30	0.28	2.75	−0.91	0.31
Best match	6.07	−27.83	1.62	−12.63	7.97	−12.63
Average	11.17	12.54	11.17	12.54	11.17	12.54

To further examine the prediction performance, we tested our approach using 30 randomly picked flights for each airport pair. Because Δt_{term} can be positive or negative for a flight, in Table 5, we report the root mean square of the 30 Δt_{term} values for the two ODs. It can be seen that as flights got closer to the destination airport (from the 25th GCD percentile to the 75th GCD percentile), the root mean square value kept decreasing for both airport pairs, meaning the prediction accuracy kept improving, as expected. (For FLT1 and FLT2, the prediction accuracy did not exactly follow a decreasing trend as the percentile increased, which may be attributed to some randomness associated with the two specific flights selected.) All the root mean squares were much smaller than those from the two baseline estimates, reaffirming the high prediction accuracy and advantages of our approach in predicting ETA_TAB.

Table 5. Root mean square of Δt_{term} values of 30 randomly picked flights for each of the two ODs using the proposed approach and two alternative estimates, at the three GCD percentiles.

	25th GCD Percentile		**50th GCD Percentile**		**75th GCD Percentile**	
	DEN-SFO	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**
Proposed approach	1.54	2.02	1.36	1.97	1.03	1.79
Best match	9.55	9.84	9.55	8.37	9.38	8.58
Average	7.88	9.68	7.88	9.68	7.88	9.68

Besides the root mean square of Δt_{term} values, which are based on the medium ETA_TAB as the point estimate, Figures 18 and 19 present the boxplots of Δt_{term} values based on ETA_TABs from multiple trajectory predictions, for each of the 30 flights. It can be seen that the different predictions for each flight actually overall yielded quite similar results, which were better than the two baseline point estimates for the vast majority of the cases.

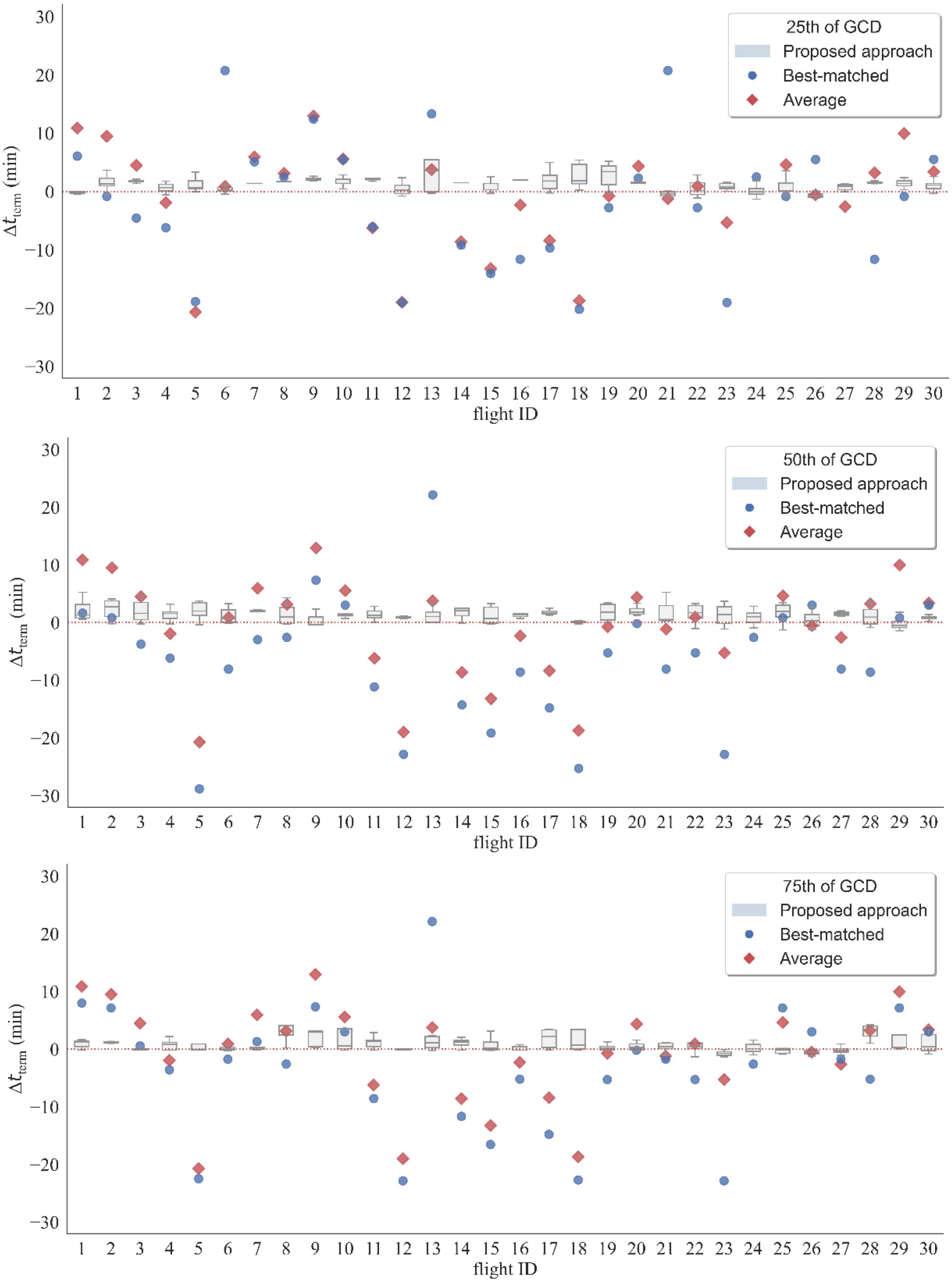


Figure 18. Boxplot of ETA_TABs using the proposed approach compared with two baseline point estimates for 30 randomly picked flights on DEN-SFO, at 25th, 50th, and 75th percentile GCD.

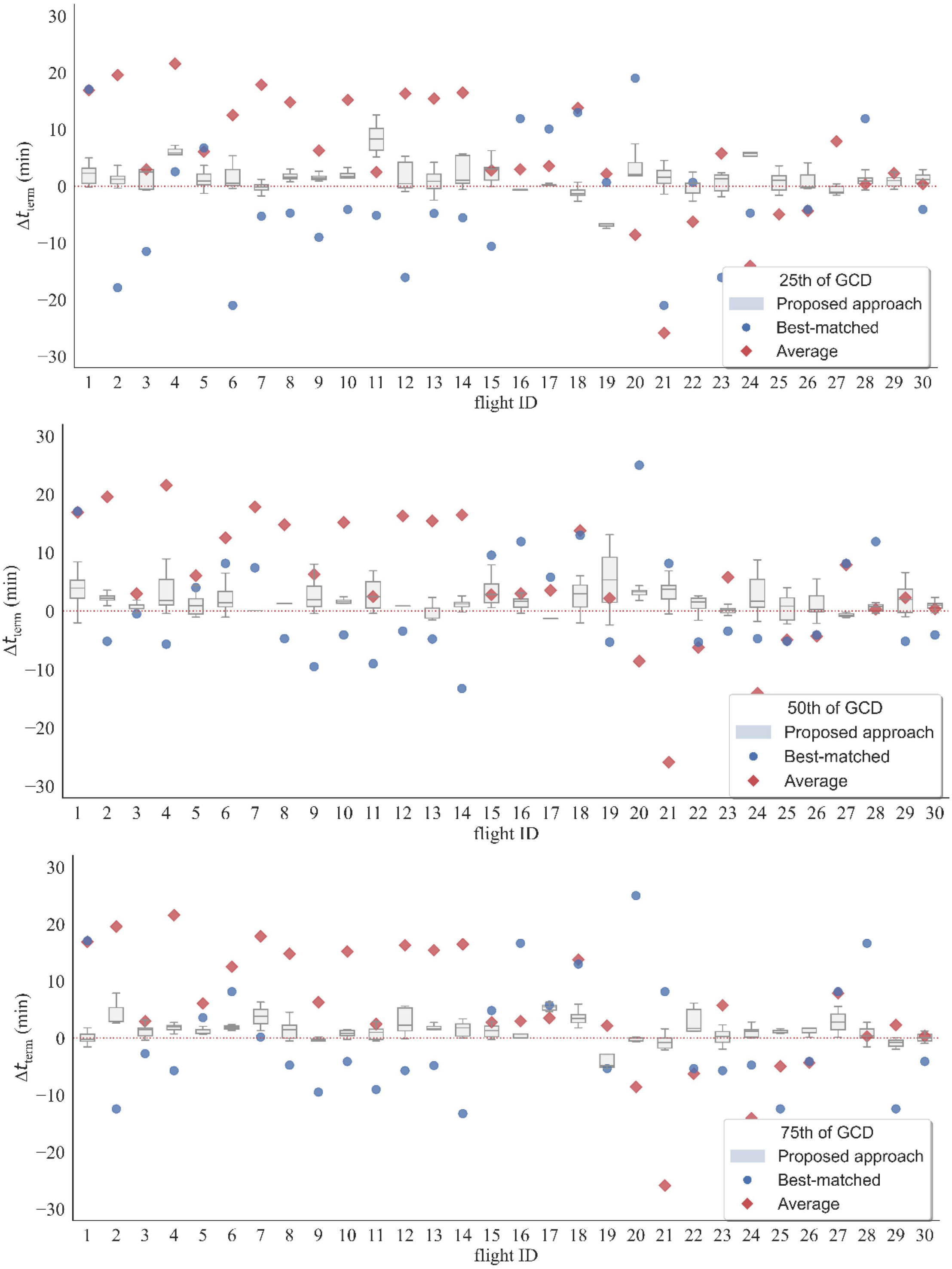


Figure 19. Boxplot of ETA_TABs using the proposed approach compared with two baseline point estimates for 30 randomly picked flights on ORD-SFO, at 25th, 50th, and 75th percentile GCD.

3.5.2. ELDT Prediction

Sampling Terminal Approach Time

As mentioned earlier, the calculation of the terminal approach time is based on performing a random draw from historical distributions. Specifically, historical terminal times were grouped according to approach entry position, aircraft type, weather condition, and airport pair, to account for their influences on the standard terminal arrival route (STAR) taken and ground speed characteristics on a particular STAR. The approach entry position of the flight under study was determined as the intersection of its best-matched trajectory and the TRACON boundary. For terminal airspace weather conditions, the flights were grouped according to whether good or bad weather, with the latter referring to having rains, fog, and/or haze, which are likely to cause poor visibility and affect flight time. Historical records of such weather conditions for the SFO terminal airspace can be accessed from the METAR weather report, which is publicly available from the National Oceanic and Atmospheric Administration (NOAA) website (https://www.aviationweather.gov/metar, accessed on 10 March 2023). The aircraft types for FLT1 and FLT2 were B777 and A320. As an illustration, the terminal approach time distributions of B777 series (including B777-200, B777-300, B777-200LR, and B777-300ER) for DEN-SFO and A320 series (including A318, A319, A320, and A321) for ORD-SFO with entry from east of SFO TRACON are shown in Figure 20 (here, the distributions include both clear and unclear days).

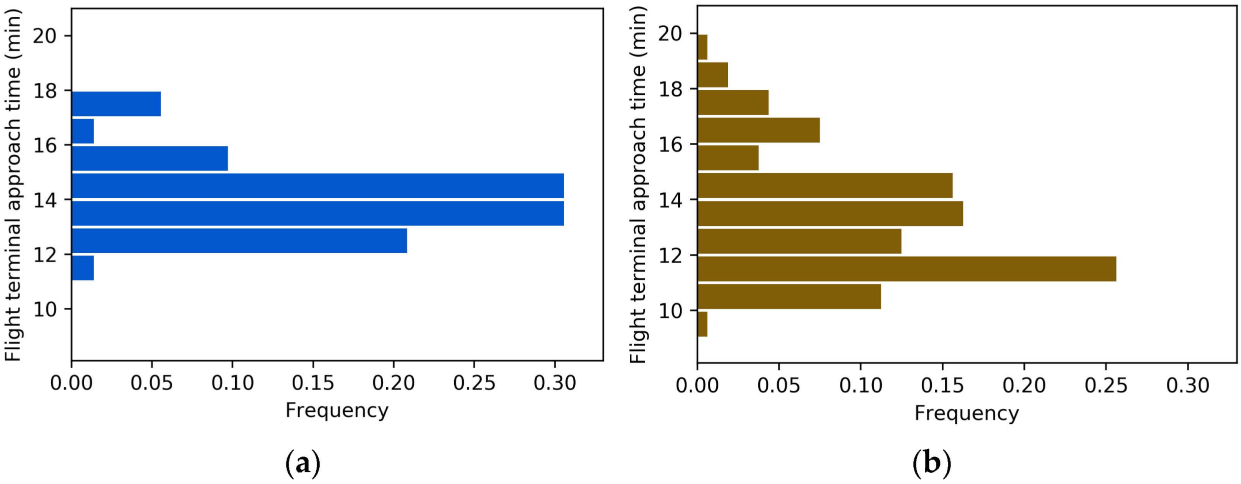


Figure 20. Illustration of flight terminal approach times: (**a**) terminal approach time of B777 series flights for DEN-SFO with entry from east of SFO TRACON; (**b**) terminal approach time of A320 series flights for ORD-SFO with entry from east of SFO TRACON.

Results

For a given flight, its ETA_TAB was added with a terminal approach time randomly drawn from the historical records in the corresponding group, as described in "Sampling Terminal Approach Time" in Section 3.5.2, to determine an ELDT. To this end, we used the weather condition for the SFO terminal airspace at the time of prediction, which can be readily accessed from the NOAA. Because of multiple ETA_TAB predictions, multiple ELDTs were obtained, forming a distribution of ELDTs for each flight. We used the median as the point prediction and compared it with six alternative estimates. First is the "best match" estimate, which was obtained by adding the airborne time of the best-matched historical flight to the actual wheels-off time. For the second estimate, the "average" estimate, ELDT was obtained by adding the average airborne time of all historical flights of the same OD, flight number, and aircraft type to the actual wheels-off time. The third and fourth estimates were based on flight plans: In one, ELDT was obtained by adding the en-route flight time from the flight plan to the actual wheels-off time, while in the other, the scheduled landing time from the flight plan was used as ELDT. Considering that our proposed approach is based on LSTM, two additional baselines using standard RNN and GRU models in place of LSTM were also produced. The prediction results are reported in Tables 6 and 7.

Table 6. Δt_{land} based on ELDT from the proposed approach and four alternative estimates, for FLT1 and FLT2 at the three GCD percentiles.

	25th GCD Percentile		50th GCD Percentile		75th GCD Percentile	
	DEN-SFO	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**
Proposed approach—LSTM	−0.98	−3.74	0.04	−0.79	−0.83	1.27
Best match	6.08	−27.70	−0.20	−10.07	7.15	−10.07
Average	11.17	16.15	11.17	16.15	11.17	16.15
Proposed approach—RNN	−1.23	−4.65	0.54	−1.67	−1.65	1.93
Proposed approach—GRU	−1.09	−4.02	0.06	−1.04	−0.98	1.31
Based on en-route flight time from the flight plan	0	7.00	0	7.00	0	7.00
Using the scheduled landing time from the flight plan	2.10	−3.00	2.10	−3.00	2.10	−3.00

Table 7. Root mean square of $\Delta \text{t}_{\text{land}}$ values of 30 randomly picked flights for each airport pair based on ELDT from the proposed approach and four alternative estimates at the three GCD percentiles.

	25th GCD Percentile		50th GCD Percentile		75th GCD Percentile	
	DEN-SFO	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**	**DEN-SFO**	**ORD-SFO**
Proposed approach-LSTM	2.78	4.76	2.65	4.54	2.08	3.96
Best match	9.23	15.04	10.68	21.36	8.69	16.54
Average	7.46	9.09	7.46	9.09	7.46	9.09
Proposed approach-RNN	3.62	6.01	3.95	5.65	2.97	5.64
Proposed approach-GRU	3.21	5.14	3.32	5.02	2.41	4.72
Based on en-route flight time in the flight plan	3.02	7.34	3.02	7.34	3.02	7.34
Using the scheduled landing time in the flight plan	28.53	10.16	28.53	10.16	28.53	10.16

The difference between the resulting ELDT and the actual landing time, termed Δt_{land}, is reported in Table 6. We repeated this for the same 30 flights for each airport pair as in Section 3.5.1 and calculated the root mean square of the Δt_{land} values, as reported in Table 7. Similar to Table 5, we observe that as flights got closer to the destination airport (from the 25th GCD percentile to the 75th GCD percentile), the root mean square value kept decreasing for both airport pairs, as expected. In addition, we found that the proposed approach with the LSTM model for trajectory prediction yielded prediction errors that were either the best or close to the best estimates for both FLT1 and FLT2 as well as the 60 randomly picked flights.

4. Summary and Further Discussion

In this paper, we proposed a data-light and trajectory-based machine learning approach for the online prediction of flight ETA while a flight is airborne, specifically the time of arrival at the destination terminal airspace boundary and landing. The basic idea of the approach was to predict the remaining trajectory and ground speed of a flight using LSTM and GBM models, respectively, and then calculate its ETA based on the predicted trajectory and speed. First, the trajectories of the current flight as well as similar historical flights were reconstructed, such that the points on a trajectory were separated by an equal distance. Based on the reconstructed trajectories, the flown trajectory of the current flight was matched with a historical trajectory with the greatest similarity. We then used historical flight trajectories to train the LSTM models, which were used to iteratively predict the remaining trajectory of the flight under study. For this, a construction layer that performed interval correction and trajectory prediction smoothing was adopted. Meanwhile, the GBM models were trained to iteratively predict flight speed along the predicted trajectory. Based on the predicted trajectory and speed, the arrival time at the destination terminal airspace boundary was estimated. We further added a randomly sampled terminal approach time from historical records to the estimated arrival time at the terminal airspace boundary, to generate an estimate of the landing time. Through numerical experiments using real-world data, we found that the proposed approach yielded better prediction performance than multiple alternative methods.

The strength of our approach lies in needing light data that are also easily accessible. Given the light and easily accessible data, our approach yields flight ETA predictions with reasonable accuracy and outperforms several alternative estimates. Nonetheless, we acknowledge that with additional data such as airspace congestion, en-route weather conditions, and air traffic control, which may not be readily available to the potential users of our proposed approach, more sophisticated models could be trained with the potential to further improve the prediction accuracy. For extensions of this approach, a few directions could be explored. First, a larger number of historical trajectories may be available and used for trajectory matching. Recall that the computation complexity of matching is $O(mnk + k\log k)$, with m being the number of flown trajectory points of the current flight, n being the maximum number of trajectory points on a historical trajectory, and k being the number of historical flight trajectories. With a larger number of historical trajectories, two possibilities could be considered to preserve computation efforts. First is reducing n, for example, by considering only a subset of points in a historical trajectory whose positions in the trajectory point sequence are similar to the position of the trajectory point under evaluation in the current flight. Second is reducing k, by clustering historical trajectories and performing matching with only one or a few representative trajectories from each cluster.

Second, while we did not include altitude in trajectory prediction, future research could explore including it as part of x_t values in LSTM. Including altitude, however, would probably demand a more elaborate characterization of the LSTM structure given the different changing patterns of latitude/longitude and altitude on a flight's trajectory. Specifically, the latitude/longitude changes are gradual, as a flight is constantly flying toward the destination airport. By contrast, a flight would stay more or less at the same altitude during the cruise and only starts to descend when near the terminal airspace (and the descent may not be continuous).

Third, research efforts can be directed toward refining the prediction of flight terminal approach time. In the current approach, the prediction is based on random sampling, although the sampling does differentiate by potential influencing factors such as aircraft type and approach entry position. Prediction methods that are more quantitative and not data-intense could be considered, for example, by developing a method that combines classification based on flight state at/around the entry of the terminal airspace and the prediction of terminal approach time in each class using neural networks.

Author Contributions: Conceptualization, Z.Z. and B.Z.; methodology, Z.Z. and B.Z.; software, Z.Z.; validation, Z.Z.; formal analysis, Z.Z. and B.Z.; investigation, Z.Z.; resources, Z.Z.; data curation, Z.Z. and W.W.; writing—original draft preparation, Z.Z. and B.Z.; writing—review and editing, Z.Z. and B.Z.; visualization, Z.Z.; supervision, B.Z., W.W. and W.T.; project administration, Z.Z. and W.W.; funding acquisition, W.T and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 71971112.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Ground Speeds of FLT1 and FLT2

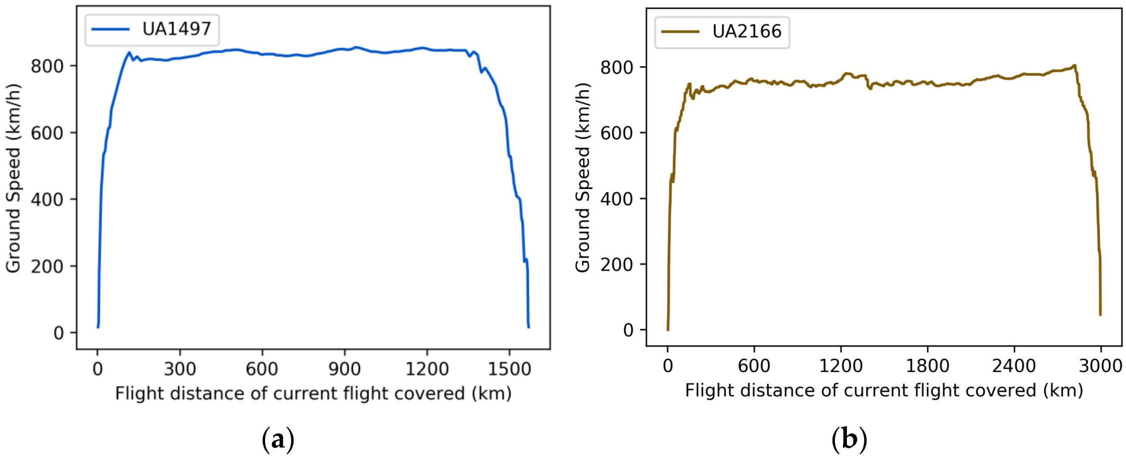


Figure A1. (**a**) Ground speed of FLT1 (DEN-SFO); (**b**) ground speed of FLT2 (ORD-SFO).

References

1. Heere, K.R.; Zelenska, R.E. A Comparison of Center/TRACON Automation System and Airline Time of Arrival Predictions. No. NAS 1.15: 209584; National Aeronautics and Space Administration, Ames Research Center: Moffett Field, CA, USA, 2000.
2. Huang, H.; Roy, K.; Tomlin, C. Probabilistic Estimation of State-Dependent Hybrid Mode Transitions for Aircraft Arrival Time Prediction. In Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head, CA, USA, 20–23 August 2007; p. 6695.
3. Flightradar24. Using the New FLIGHTRADAR24 KML and CSV Export Tools. 2020. Available online: https://www.flightradar24.com/blog/using-the-new-flightradar24-kml-and-csv-export-tools/ (accessed on 25 August 2020).
4. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
5. Graves, A.; Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), Beijing, China, 21–26 June 2014; pp. 1764–1772.
6. Chorowski, J.; Bahdanau, D.; Cho, K.; Bengio, Y. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv* **2014**, arXiv:1412.1602.
7. Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A.C.; Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems 28*; Curran Associates Inc.: Red Hook, NY, USA, 2015; pp. 2980–2988.
8. Sutskever, I.V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*; Curran Associates Inc.: Red Hook, NY, USA, 2014; pp. 3104–3112.
9. Karpathy, A.; Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3128–3137.
10. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 652–663. [CrossRef] [PubMed]
11. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232.
12. Ma, X.; Ding, C.; Luan, S.; Wang, Y.; Wang, Y. Prioritizing influential factors for freeway incident clearance time prediction using the gradient boosting decision trees method. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 2303–2310. [CrossRef]
13. Ding, C.; Cao, X.J.; Næss, P. Applying gradient boosting decision trees to examine non-linear effects of the built environment on driving distance in Oslo. *Transp. Res. Part A Policy Pract.* **2018**, *110*, 7–117. [CrossRef]
14. Barua, L.; Zou, B.; Liu, Y. Modeling Household Online Shopping Demand in the US: A Machine Learning Approach and Comparative Investigation between 2009 and 2017. *arXiv* **2021**, arXiv:2101.03690.
15. Barua, L.; Zou, B. Planning maintenance and rehabilitation activities for airport pavements: A combined supervised machine learning and reinforcement learning approach. *Int. J. Transp. Sci. Technol.* **2022**, *11*, 423–435. [CrossRef]
16. Krozel, J.L. Estimating time of arrival in heavy weather conditions. In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Portland, OR, USA, 9–11 August 1999; p. 4232.
17. Bai, X.; Weitz, L.A.; Priess, S. Evaluating the impact of estimated time of arrival accuracy on interval management performance. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego, CA, USA, 4–8 January 2016; p. 1852.
18. Franco, A.; Rivas, D.; Valenzuela, A. Probabilistic aircraft trajectory prediction in cruise flight considering ensemble wind forecasts. *Aerosp. Sci. Technol.* **2018**, *82*, 350–362. [CrossRef]
19. Zhang, J.; Liu, J.; Hu, R.; Zhu, H. Online four dimensional trajectory prediction method based on aircraft intent updating. *Aerosp. Sci. Technol.* **2018**, *77*, 774–787. [CrossRef]
20. Shi, Z.; Xu, M.; Pan, Q. 4-D Flight trajectory prediction with constrained LSTM network. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 7242–7255. [CrossRef]

21. Ayhan, S.; Costas, P.; Samet, H. Predicting estimated time of arrival for commercial flights. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 33–42.
22. De Leege, A.; van Paassen, M.; Mulder, M. A machine learning approach to trajectory prediction. In Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, MA, USA, 19–22 August 2013; p. 4782.
23. Wang, Z.; Liang, M.; Delahaye, D. A hybrid machine learning model for short-term estimated time of arrival prediction in terminal manoeuvring area. *Transp. Res. Part C Emerg. Technol.* **2018**, *95*, 280–294. [CrossRef]
24. Wang, Z.; Liang, M.; Delahaye, D. Automated data-driven prediction on aircraft Estimated Time of Arrival. *J. Air Transp. Manag.* **2020**, *88*, 101840. [CrossRef]
25. Glina, Y.; Jordan, R.; Ishutkina, M. A tree-based ensemble method for the prediction and uncertainty quantification of aircraft landing times. In Proceedings of the American Meteorological Society–10th Conference on Artificial Intelligence Applications to Environmental Science, New Orleans, LA, USA, 22–26 January 2012.
26. Kim, M.S. Analysis of short-term forecasting for flight arrival time. *J. Air Transp. Manag.* **2016**, *52*, 35–41. [CrossRef]
27. Achenbach, A.; Spinler, S. Prescriptive analytics in airline operations: Arrival time prediction and cost index optimization for short-haul flights. *Oper. Res. Perspect.* **2018**, *5*, 265–279. [CrossRef]
28. Lui, G.N.; Klein, T.; Liem, R.P. Data-Driven Approach for Aircraft Arrival Flow Investigation at Terminal Maneuvering Area. In Proceedings of the AIAA AVIATION 2020 Forum, Online, 15–19 June 2020; p. 2869.
29. Liu, Y.; Hansen, M. Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach. *arXiv* **2018**, arXiv:1812.11670.
30. Zhang, X.; Mahadevan, S. Bayesian neural networks for flight trajectory prediction and safety assessment. *Decis. Support Syst.* **2020**, *131*, 113246. [CrossRef]
31. Shi, Z.; Xu, M.; Pan, Q.; Yan, B.; Zhang, H. LSTM-based flight trajectory prediction. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
32. OpenFlight. Retrieved from Airport, Airline and Route Data. Available online: https://openflights.org/data.html (accessed on 15 May 2022).
33. Olah, C. Understanding LSTM Networks. Available online: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed on 22 January 2021).
34. Movable Type Ltd. Calculate Distance, Bearing and more between Latitude/Longitude Points. Available online: https://www.movable-type.co.uk/scripts/latlong.html (accessed on 26 August 2020).
35. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*; Springer: New York, NY, USA, 2009.
36. Murça, M.C.R.; Hansman, R.J.; Li, L.; Ren, P. Flight trajectory data analytics for characterization of air traffic flows: A comparative analysis of terminal area operations between New York, Hong Kong and Sao Paulo. *Transp. Res. Part C Emerg. Technol.* **2018**, *97*, 324–347. [CrossRef]
37. FAA. Amendment of Class B Airspace; San Francisco, CA, USA. 2018. Available online: https://www.federalregister.gov/documents/2018/06/08/2018-12304/amendment-of-class-b-airspace-san-francisco-ca (accessed on 9 October 2020).
38. FAA. Instrument Procedures Handbook. 2017. Available online: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/instrument_procedures_handbook/media/FAA-H-8083-16B.pdf (accessed on 9 October 2020).
39. Chollet, F. Keras. 2015. Available online: https://github.com/fchollet/keras (accessed on 14 September 2020).
40. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.