Decision Support Systems 131 (2020) 113246

Contents lists available at ScienceDirect

Decision Support Systems

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

Bayesian neural networks for flight trajectory prediction and safety assessment☆

Xiaoge Zhang, Sankaran Mahadevan*

Department of Civil and Environmental Engineering, School of Engineering, Vanderbilt University, Nashville, TN 37235, USA

ARTICLE INFO

Keywords:
Deep learning
Big data
Trajectory prediction
Air transportation system
Bayesian neural network
Safety assessment

ABSTRACT

Safety, as the most important concern in civil aviation, needs to be maintained at an acceptable level at all times in the air transportation system. This paper aims to increase en-route flight safety through the development of deep learning models for trajectory prediction, where model prediction uncertainty is characterized following a Bayesian approach. The proposed methodology consists of four steps. In the first step, a large volume of raw messages in Flight Information Exchange Model (FIXM) format streamed from Federal Aviation Administration are processed with a distributed computing engine Apache Spark to extract trajectory information in an efficient manner. In the second step, two types of deep learning models are trained to predict flight trajectory from different perspectives. Specifically, deep feedforward neural networks (DNN) are trained to make a one-step-ahead prediction on the deviation along latitude and longitude between the actual flight trajectory and target flight trajectory. In parallel, deep Long Short-Term Memory (LSTM) neural networks are trained to make longer-term predictions on the flight trajectory over several subsequent time instants. The DNN model is more accurate but has a single-step prediction horizon, whereas the LSTM model is less accurate but longer prediction horizon. Therefore, in the third step, the two different types of deep learning models are blended together to create a multi-fidelity prediction. After quantifying the discrepancy between the two model predictions in the current time instant, the DNN prediction is used to correct the LSTM prediction of flight trajectory along subsequent time instants accordingly. The multi-fidelity approach is expanded to multiple flights, and is then used to assess safety based on horizontal and vertical separation distance between two flights. Computational results illustrate the promising performance of the blended model in predicting the flight trajectory and assessing en-route flight safety.

1. Introduction

As forecast by the International Air Transport Association (IATA), the worldwide air travel demand will be nearly doubled over the next two decades compared to the 3.8 billion air travellers in 2016 [1, 2]. One straightforward way to meet the fast-growing air travel demand is to increase the flight density by reducing the minimum horizontal and vertical separation distance between two adjacent airplanes. By doing this, the same airspace could afford more airplanes than before, but of course this increases congestion in the airspace. Since safety is of the greatest importance in the field of civil aviation, it is critical to investigate whether we are able to maintain the safety of the air transportation system at an acceptable level before implementing the strategic separation distance reduction. For system safety assurance, accurate and robust predictive models need to be developed to assess the en-route flight safety over time, in which the uncertainties arising from heterogeneous sources should be accounted, e.g., Automatic Dependent Surveillance-Broadcast (ADS-B) transponder precision, weather forecast uncertainty, and model predictive uncertainty.

At the forefront of transformation to future National Airspace System (NAS), an important upgrade in the Next Generation Air Transportation System (NextGen) is the transition from radar-based operations to satellite-based navigation and surveillance. Together with ADS-B system, satellite-based operations provide enhanced accuracy for

☆ **List of Abbreviations**: **IATA:**International Air Transport Association, **ICAO:**International Civil Aviation Organization, **NAS:**National Airspace System, **ADS-B:**Advanced Dependent Surveillance-Broadcast, **ATO:**Air Traffic Organization, **NextGen:**Next Generation Air Transportation System, **FAA:**Federal Aviation Administration, **SWIM:**System-Wide Information Management, **ATC:**Air Traffic Control, **ATM:**Air Traffic Management, **BADA:**Base of Aircraft Data, **SFDPS:**SWIM Flight Data Publication Service, **FIXM:**Flight Information Exchange Model, **ARTCC:**Air Route Traffic Control Center, **RAE:**Risk Analysis Event.

* Corresponding author.
E-mail addresses: zxgcqupt@gmail.com (X. Zhang), sankaran.mahadevan@vanderbilt.edu (S. Mahadevan).

https://doi.org/10.1016/j.dss.2020.113246
Received 18 April 2019; Received in revised form 25 November 2019; Accepted 2 January 2020
Available online 13 January 2020

aircraft navigation, surveillance, and position tracking [3, 4]. With the centralized NAS data sharing backbone – System Wide Information Management (SWIM) – real-time, accurate flight, surveillance, weather, and aeronautical information is broadcast to relevant stakeholders on a regular basis. As indicated in Federal Aviation Administration (FAA)'s report on the future of the NAS [5], one crucial need in NextGen is to develop predictable and efficient services across tower, terminal, and en-route domains, which can then be used to assist controllers to handle off-nominal events or demand-capacity imbalances in a more strategic and efficient manner. As shown in the report from the air traffic organization (ATO) at FAA, there are 7249 validated loss of separation cases within U.S. airspace in fiscal year 2015 [6]. Compared to FY14, FY15 saw a 5% increase in the total number of risk analysis events (RAEs). Given the projected air traffic growth forecast by IATA, if no operation is taken considering increased air traffic density, the number of separation loss cases will be more than doubled in two decades. Towards this end, flight trajectory prediction is an important need; accurate prediction of flight trajectory is vital to avoid accidents and reduce errors while ensuring safety and efficiency.

In brief, flight trajectory is the core information that is used by the system as a basis for distributing flight information to relevant airlines and air traffic control (ATC) units, facilitating timely coordination between sectors and units, correlating flight data with tracks, monitoring the adherence of an aircraft with its assigned route, and detecting and resolving conflicts. In this respect, trajectory prediction algorithms are a crucial component of decision support tools (DST) for conflict detection and resolution, arrival metering, and other applications in air traffic management automation. Considering its paramount importance in assuring the safety of air transportation system, a large number of studies have focused on the development of models and algorithms for trajectory prediction (TP) over the past ten years [7]. For example, Prats et al. [8] used an iterative quasi-Newton method to find departure flight trajectory that minimizes the noise annoyance. Leege et al. [9] trained a machine learning model to predict flight arrival time over points along the fixed arrival route, where aircraft state parameters (i.e., aircraft type, aircraft ground speed) and meteorological conditions (i.e., surface wind, altitude wind) were used as model inputs. Gong and McNally [10] developed a versatile methodology for automated trajectory prediction analysis such that the error measurements were sensitive to small trajectory prediction algorithm changes.

The current state-of-the-art approaches for trajectory prediction can be grouped into two major categories: physics-based models and data-driven models. In physics-based models, the aircraft is represented as a point mass, and Newton's law is utilized to associate the force/thrust generated by the aircraft engine with the inertial acceleration of its mass [11]. Typically, physics-based models are represented as a set of differential equations; given the initial state of an aircraft (i.e., mass, thrust, position, velocity, bank angle) and meteorological conditions (i.e., wind velocity and direction), we predict the successive points of future aircraft trajectory by integrating the differential equations over a time interval. Take the Base of Aircraft Data (BADA) as an example; it is an aircraft performance model designed with a mass-varying, kinetic approach as developed and maintained by EUROCONTROL [12], which is used for trajectory computation, simulation and prediction in Air Traffic Management (ATM). However, physics-based models typically require explicit modeling of aircraft performance procedures and the real-time aircraft state (i.e., thrust, speed intent), while the models and most of model inputs might not be always readily available to ground-based systems due to commercial sensitivity. Considering this significant hurdle in physics-based models, researchers have also considered data-driven models for predicting the flight trajectory by learning from the historical flight trajectories with machine learning and data mining algorithms [13-15]. A large amount of flight trajectories are available, which makes it possible to mine the complex trajectory patterns and extract important features. For example, Alligier and Gianazza [16] applied a machine learning approach to predict aircraft mass and speed intent on the climb phase with the ADS-B data from the OpenSky network. Di Ciccio et al. [17] developed an automated prediction model to detect flight diversions based on the flight track updates (i.e., position, velocity, altitude, and intended destination) such that the receiving parties could respond in a timely way to unexpected events that occurred during the flight. Among the machine learning models, deep learning (or deep neural network) has gained popularity due to its promising features in automatic feature extraction and end-to-end modeling [18, 19]. For example, Moradi Kordmahalleh et al. [20] developed a sparse recurrent neural network with flexible topology to make trajectory prediction 6 and 12 h ahead of four catastrophic Atlantic hurricanes. Alahi et al. [21] proposed an LSTM-based recurrent neural network to jointly reason across multiple individuals to predict human trajectories in an area, in which one LSTM was trained for each trajectory and the information was shared between the LSTMs through the introduction of a new social pooling layer.

If the model is trained with limited or noisy data, model prediction uncertainty is an important consideration in machine learning [22-24]. In particular, in the aviation sector, it is extremely important to account for trajectory prediction uncertainty as it is closely related to risk assessment and decision making w.r.t. quantities pertaining to flight safety [25], e.g., separation distance. In this paper, we characterize model prediction uncertainty from a Bayesian perspective. Bayesian probability theory provides a mathematically well-grounded approach to reason about model prediction uncertainty. However, given the large number of parameters to be optimized in deep learning, the implementation of Bayesian inference in deep learning is usually unachievable due to the unaffordable computational effort. Recently, Gal and Ghahramani [26, 27] proposed that dropout can be interpreted as a variational approximation to the posterior of a Bayesian neural network. More importantly, the variational approximation to Bayesian inference with dropout reduces computational complexity without sacrificing model prediction accuracy. As a result, dropout implementation to approximate Bayesian inference has drawn tremendous attention in many applications, such as forecasting the number of Uber trips [28], and disease detection [29].

In this paper, we train two separate Bayesian neural networks to make flight trajectory predictions. Considering the massive flight tracking messages in FIXM format streamed from the FAA SWIM flight data publication service (SFDPS), we leverage a robust distributed computing platform – Apache Spark – to parse the large volume of raw data [30, 31], from which the data pertaining to actual flight trajectory, target flight trajectory, aircraft states (e.g., velocity, altitude, and position), as well as historical flight trajectory is extracted. Next, we build two different types of Bayesian neural network models: in the first model, deep feedforward Bayesian neural networks (DNN) are trained with historical data for one-step-ahead prediction on the deviation between actual trajectory and target flight trajectory. In the second model, LSTM neural networks are trained with historical flight trajectory data to make longer-term predictions on the future flight trajectory. Afterwards, the two models are blended through a discrepancy term to make accurate trajectory prediction for ongoing flights. A separation distance-based safety indicator is developed to measure flight safety between two flights in the air transportation system. The major contributions we have made in this paper are multifold.

1. We develop a program with a high performance computing platform – Apache Spark – to parse the raw SFDPS data in FIXM format, which creates the inputs to Bayesian deep learning models.
2. Two types of Bayesian neural networks are trained to make flight trajectory prediction from different perspectives. In both models, model prediction uncertainty is quantified with the dropout strategy, and propagated through the integrated model with Monte Carlo samples.
3. The two types of Bayesian neural networks are integrated for making accurate long-term predictions for ongoing flights. By doing

this, we leverage the advantages of both models: the high prediction accuracy of the DNN model and longer-term prediction capability of the LSTM model.
4. A probabilistic safety indicator is used to measure the horizontal and vertical separation distance between two flights.

The rest of the paper is structured as follows. Section 2 provides a brief introduction to the SWIM data source, and describes the major components in each record. Section 3 introduces the concept of Bayesian neural networks. Section 4 develops two probabilistic deep learning models to predict the trajectory of an ongoing flight. Section 5 evaluates the performance of the blended machine learning model on a test dataset, and demonstrates its usefulness in updating the in-flight safety indicator. Section 6 provides concluding remarks and discusses future research directions.

2. SWIM SFDPS data overview

SWIM is a NAS-wide information system that supports NextGen goals. Acting as a digital data-sharing backbone, SWIM enables increased common situational awareness and improved NAS agility to deliver the right information to the right people at the right time [32]. Users with granted credentials from FAA have access to the SWIM database via subscription to particular channels (i.e., SWIM Terminal Data Distribution System — STDDS, Time Based Flow Management — TBFM, SWIM Flight Data Publication Service — SFDPS, and others). In this paper, we focus on analyzing flight trajectory data by subscribing to the SFDPS stream. In brief, SFDPS provides real-time en-route flight data via batched track messages and custom FIXM formats to NAS consumers.

Fig. 1 illustrates a specific SFDPS tracking message for a flight operated by the JetBlue airline that departed from the Boston Logan International Airport (code: KBOS) and arrived at the Baltimore–Washington International Airport (code: KBWI) on October 16, 2018. The en-route flight tracking information is stored in the International Civil Aviation Organization (ICAO) FIXM format. There are five major types of information in each tracking message: flight origin and destination, timestamp, en-route flight state (i.e., flight speed and velocity, altitude, position, target altitude and position), flight ID and supplemental data. In SFDPS, flight tracking information is broken out by flight with just one track update per flight in one message. The flight tracking messages are updated on a regular basis of 12 s for an active flight while the flight is within an Air Route Traffic Control Center's (ARTCC) airspace. Since there are more than 20,000 commercial flights within the U.S. airspace every day, we stream nearly 100 GB of SFDPS messages from SWIM per day. Given such big data, an efficient method is needed to parse the flight trajectory data from the massive FIXM files.

3. Deep learning

In this section, we briefly describe the underlying mechanisms in the feedforward neural network and LSTM neural network. Afterwards, a generic framework with Monte Carlo dropout is introduced to approximate Bayesian inference in deep neural networks. The feedforward neural network and LSTM neural network are leveraged to develop two individual models to predict flight trajectories in Section 4, where Monte Carlo dropout is used to quantify the uncertainty in the prediction made by the two deep learning models.

3.1. Feedforward neural network

Feedforward neural networks are artificial neural networks where the connections between units do not form a cycle. In general, there are four major components in a feedforward neural network: neuron, activation function, cost function, and optimization. Fig. 2 shows a feedforward neural network consisting of two hidden layers with each hidden layer having four neurons. In the feedforward neural network, the input layer receives the values of input variables, then the sum of

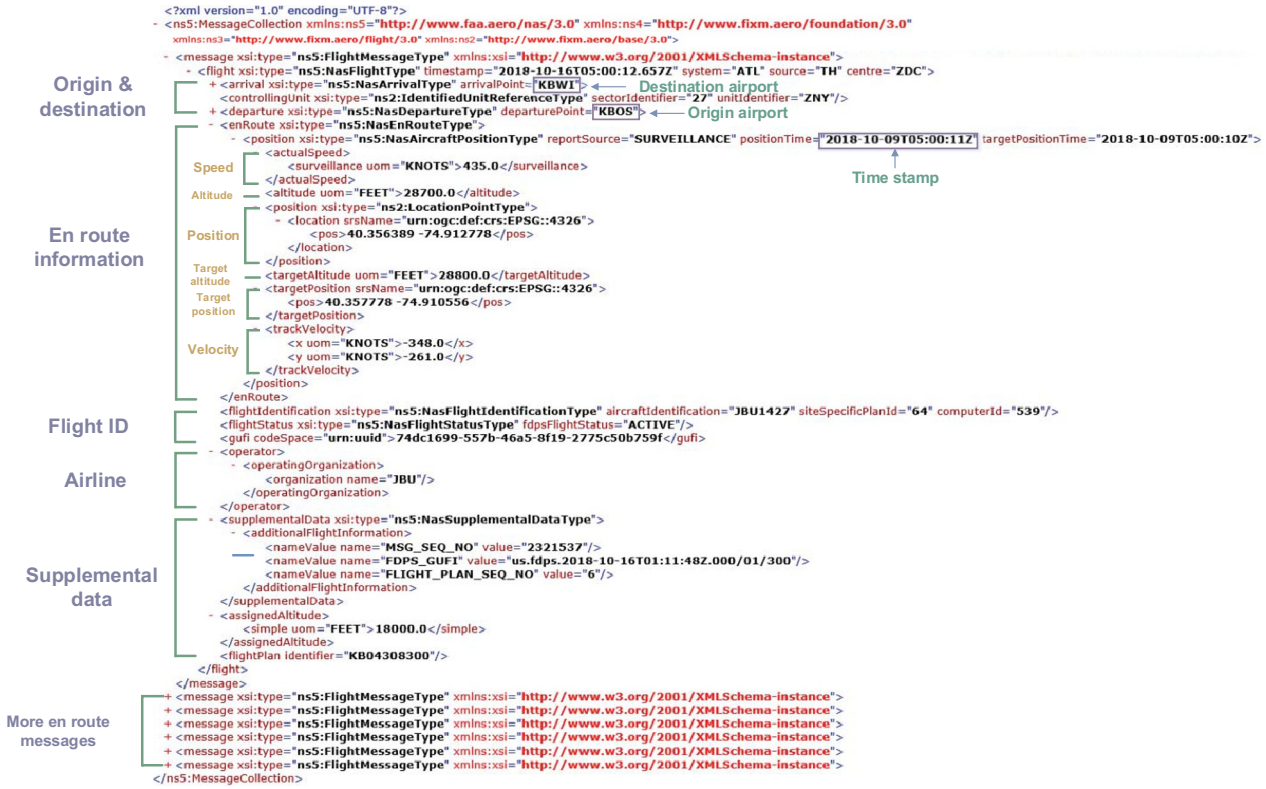


Fig. 1. An illustration of the SFDPS en-route flight tracking message.

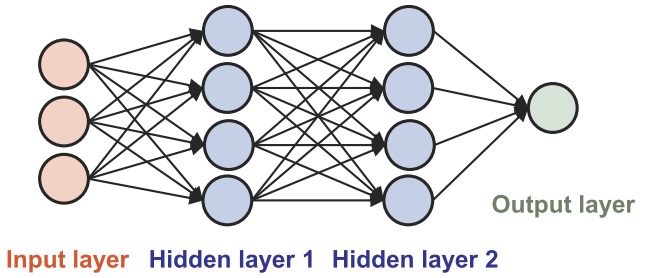


Fig. 2. A feedforward neural network with two hidden layers [33].

the products of the neuron weights and the inputs is performed at each neuron. Next, the summed value is passed through an activation function, which squashes the summed value into a fixed range.

Following the similar way, input variables are passed through a stack of hidden layers. The predictive value from the output layer of neural network is compared against the actual value. The loss function is defined to measure how far off the feedforward neural network predictions are away from the observations for the training samples. Afterwards, backpropagation algorithms calculate the error at the output and then distribute it back through the network layers using chain rule to compute gradients for each layer [34]. Next, adjustments are applied to the weights in each layer of the feedforward network so as to minimize the loss value in the next iteration. The same procedures continue until the loss function converges to a stable value.

3.2. Long short-term memory neural network

Different from the feedforward DNN, a recurrent neural network (RNN) contains a feedback loop that is connected to the past decisions (e.g., the information contained in the hidden cell) and ingests its own output moment after moment as input, which enables the network to learn the temporal dependencies in time series (or sequence) data. By preserving sequential information and passing information from one step to the next through the hidden state variables, RNN manages to maintain a memory of the temporal correlation between events separated by many moments. This feature makes it ideal to handle time series data. Among RNNs, the LSTM, as proposed by Hochreiter and Schmidhuber [35], has received increasing attention because it overcomes the problem of vanishing and exploding gradients by enabling the network to learn when to forget the previous hidden states and when to update the hidden states by integrating with new memory in an adaptive manner [35].

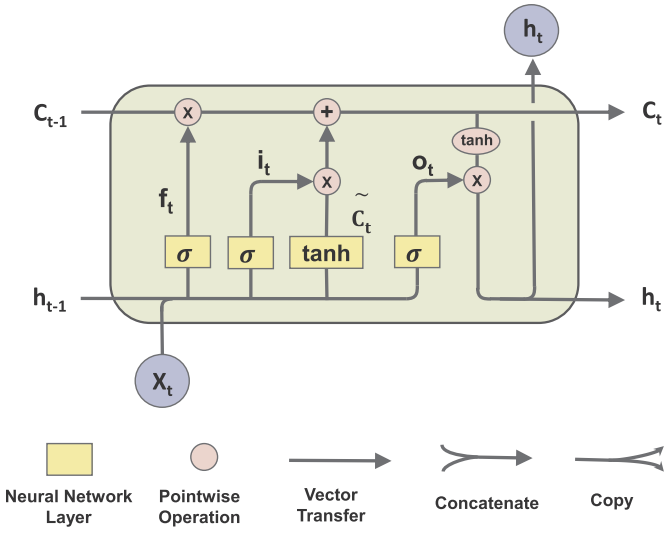


Fig. 3. Structure of an LSTM block [36].

Fig. 3 shows a graphical representation of all the mathematical operations going on within an LSTM block. Let $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ denote a general input sequence for an LSTM, where $\boldsymbol{x}_t$ represents a k-dimensional input vector at the t-th time step. The most important element in an LSTM block is the cell state as represented by the horizontal line running through the top of the diagram in Fig. 3. The first step in the LSTM block is to decide what information to forget, which is determined by a sigmoid layer called "forget gate layer" as denoted in Eq. (1a). The output of the forget gate layer is a vector of the same length, as there are a number of neurons with each element of the vector being between 0 and 1, where 0 indicates the information in the previous cell state $\boldsymbol{C}_{t-1}$ is completely thrown away, and 1 reveals that the information in the previous cell state is completely retained. In the next step, decision is made regarding what new information to store in the cell state. In this case, a sigmoid "input gate layer" is firstly used to determine what values to update (see Eq. 1b), then a tanh layer creates a set of candidate values $\tilde{\boldsymbol{C}}_t$ to be added to the new cell state (see Eq. 1c). Next, the new cell state is updated by combining the information retained in the previous cell state $\boldsymbol{C}_{t-1}$ with the new candidate information $\tilde{\boldsymbol{C}}_t$ to be added (see Eq. 1d). Finally, the output of the LSTM cell is decided. To achieve this goal, a sigmoid layer is first used to determine which part to output (see Eq. 1e), then the new cell state $\boldsymbol{C}_t$ is pushed through a tanh layer such that the value is squashed into the range $[-1, \quad 1]$. The product of $\boldsymbol{o}_t$ and $\tanh(\boldsymbol{C}_t)$ composes the output value (see Eq. 1f).

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \\
\tilde{\boldsymbol{C}}_t &= \tanh(\boldsymbol{W}_C[\boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}_C) \\
\boldsymbol{C}_t &= \boldsymbol{f}_t \times \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \times \tilde{\boldsymbol{C}}_t \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \times \tanh(\boldsymbol{C}_t)
\end{aligned} \tag{1f}
$$

where $\boldsymbol{W}_f$, $\boldsymbol{W}_i$, $\boldsymbol{W}_C$, and $\boldsymbol{W}_o$ represent the weight matrices associated with each unit, and $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_C$, and $\boldsymbol{b}_o$ are the bias terms related to each unit.

The above description briefly introduces the underlying mechanism within a single LSTM cell. In practice, a number of LSTM blocks can be concatenated together in a hidden layer, and several hidden layers can be stacked together to learn more complex patterns from time series data.

3.3. Bayesian neural network

Both feedforward neural network and LSTM neural network can be considered as regression models in general, and these models are trained to relate a series of inputs $\boldsymbol{X} = \{X_1, \quad X_2, \cdots, X_N\}$ with their corresponding outputs $\boldsymbol{Y} = \{Y_1, \quad Y_2, \cdots, Y_N\}$. Suppose we have a neural network denoted as $f^{\omega}(\cdot)$, where f represents the structure of the neural network (e.g., number of layers and hidden units, choice of activation functions), and $\boldsymbol{\omega}$ is the collection of model parameters (neuron weights) to be estimated. In the Bayesian context, we aim to infer the posterior distribution of the model parameters $\boldsymbol{\omega}$ that are most likely to generate the observed data X and Y. In the Bayesian approach, we assign a prior distribution over the space of the model parameters $p(\boldsymbol{\omega})$ to represent our prior belief on each candidate model parameter in generating the observed data, then a likelihood function $p(Y_i|\, X_i, \omega)$ ($\omega \in \boldsymbol{\omega}$) is constructed to characterize the probability of generating the observed data given model parameter ω. Bayesian inference aims to compute the posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$ over the model parameters $\boldsymbol{\omega}$; this computation can be formulated as below.

$$
\overset{\text{Posterior}}{p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})} = \frac{\overset{\text{Likelihood}}{p(\boldsymbol{Y}|\, \boldsymbol{X}, \omega)}\overset{\text{Prior}}{p(\omega)}}{p(\boldsymbol{Y}|\, \boldsymbol{X})} \tag{2}
$$

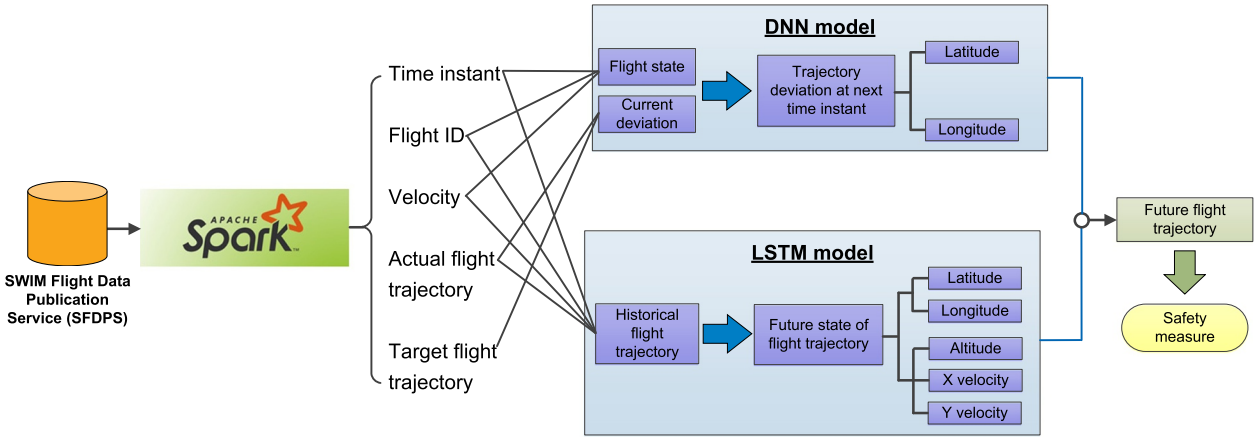


Fig. 4. Flowchart of the proposed methodology for flight trajectory prediction.

Once we have the full posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$, the predictive distribution on a new data point x^* can be obtained by integrating over $\boldsymbol{\omega}$:

$$p(y^*|\, x^*, \boldsymbol{X}, \boldsymbol{Y}) = \int p(y^*|\, x^*, \omega) p(\omega|\, \boldsymbol{X}, \boldsymbol{Y}) d\omega \tag{3}$$

where $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$ denotes the posterior distribution of $\boldsymbol{\omega}$. In practice, we draw a sample ω from the posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$ each time, and obtain a point estimate y^* on the new data point x^* for the given ω. The aforementioned operations are repeated many times, from which we obtain a distribution estimate on the new data point x^*.

With respect to neural networks, a popular way to define the prior distributions is to place a Gaussian distribution $p(\omega) = \mathcal{N}(0, I)$ over the neuron weights while the bias vectors are often assumed to be point estimates for the sake of simplicity [37]. Unfortunately, it is computationally intractable to perform Bayesian inference with the classical Monte Carlo Markov Chain (MCMC) sampling algorithms (e.g., Gibbs sampling, Metropolis-Hastings, etc.) due to the non-linearity and non-conjugacy in deep neural networks. The non-linearity in the activation functions themselves and their combinations across multiple hidden layers result in highly non-linear behavior in the neural network. Due to the lack of conjugate prior caused by the complex structure of the neural networks, it is not possible to derive the posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$ in a closed form when performing Bayesian inference in neural networks.

Instead of inferring the exact posterior distribution, approximate inference techniques, such as variational inference, have been proposed to mitigate the computational burden [38]. The key idea in variational inference is to use a variational parametric distribution to approximate the posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$. Gal and Ghahramani [27] showed that the objective of Monte Carlo (MC) dropout, in effect, minimizes the Kullback–Leibler divergence between the approximate distribution $q_\theta(\omega)$ and the posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$, thus it can be used to approximate Bayesian inference in deep neural networks. The approach is briefly described here, and mathematical details are provided in Appendix A.

In a neural network, the term "dropout" refers to dropping out both hidden and visible units. The dropped unit along with its incoming and outgoing connections is temporarily removed from the network. The choice of which unit to drop is randomly determined. Specifically, for each training point in a mini-batch, forward and backpropagation is only done once on a thinner neural network by randomly dropping out units to calculate the gradients of neuron weights. The gradients of each parameter are then averaged over the training cases in the mini-batch to obtain the weights for the overall network. Any training case which does not use a parameter contributes a gradient of zero for that parameter. During testing, we generate random samples following a Bernoulli distribution for each unit in the input layer and the hidden layers, where each sample takes the value of 1 with a given probability of p_i. If the value of the binary variable is zero, then the corresponding unit and its incoming and outgoing connections are dropped. With the trained neural network, we construct a predictive distribution and quantify the uncertainty in our prediction on any new data point. Although the approximation of a Bayesian neural network via the MC dropout strategy is computationally efficient, there has been considerable discussion on whether this approach should be called Bayesian because it does not explicitly perform Bayesian inference with prior and posterior distributions [39].

In comparison with other state-of-the-art approaches, the MC dropout strategy has several promising features. First of all, it is very straightforward to implement. Specifically, MC dropout is applied before each hidden layer; then the model output is a collection of random samples generated from the posterior predictive distribution [40]. Secondly, MC dropout can be used to quantity the model prediction uncertainty in a variety of neural networks, such as feedforward deep neural networks, recurrent neural networks, and convolutional neural networks.

4. Proposed method

In this section, a four-step approach is developed to train two individual Bayesian deep learning models from massive historical flight trajectories to support the assessment of flight safety. The overall framework of the proposed methodology is illustrated in Fig. 4. In the first step, we develop an algorithm using a distributed high-performance computing framework – Apache Spark – to process a large volume of raw flight tracking messages in FIXM format. After the information pertaining to flight trajectory is extracted from the raw data, two different types of Bayesian deep learning models are trained to make trajectory predictions of ongoing flights from different perspectives. Specifically, DNN models are trained with historical data to forecast the deviation between the filed flight plan and the actual flight trajectory along latitude and longitude; and LSTM models are trained with historical trajectory data for making predictions on the future flight trajectory, such as latitude and longitude, altitude and velocity. Next, the predictions from the two types of deep learning models are fused together to enhance the model prediction accuracy. Finally, the fused model is then utilized to make trajectory predictions on multiple flights and evaluate the safety of two given flights, where separation distance is used as a quantitative safety metric.

4.1. SFDPS message processing

As mentioned earlier, SFDPS publishes a wide variety of en-route messages, such as flight data, airspace data, operational data, and other

general messages. Among all the published messages, we are most interested in the information relevant to flight trajectory including flight ID, origin and destination, and en-route position surveillance data (e.g., latitude, longitude, and altitude). Within the U.S. airspace, there are more than 20,000 commercial flights every day. In terms of data volume, more than 100 GB of SFDPS messages is broadcast by SWIM and streamed to the local storage disk. As shown in Fig. 1, all the significant content is batched in FIXM format. To meet the intensive computational requirements of massive data analysis, we leverage a general-purpose scalable high-performance computing framework – Apache Spark – to parse the large volume of SFDPS messages in FIXM format. As a unified engine for big data analytics, Apache Spark combines a core engine for distributed computing with an advanced programming model for in-memory data processing through a data sharing schema — resilient distributed dataset (RDD). The Spark framework's in-memory programming model results in up to 100 times faster than Hadoop in big data analytics [41]. Besides, Apache Spark offers rich APIs in high-level languages (e.g., Python, Java, Scala) for performing complex data operations, such as data transformation and querying [42]. Last but not the least, Apache Spark supports operating on a rich set of data sources through the DataFrame interface, such as JSON, CSV, database connection, and XML etc.

In this paper, we utilize the spark-xml library for parsing and querying FIXM data within Apache Spark [43]. After the raw FIXM data is parsed, we designate the row tag as 'flight', then spark-xml library transforms the content within this row tag into a row consisting of all the attribute names and their values. Next, we filter out flight tracking information that contains track updates for individual flights from all the SFDPS messages by specifying the source of tracking messages as 'TH' (see Fig. 1), then we group the trajectory data by flight date and flight ID, and rank them according to the position time (see Fig. 1 for more details). The sorted flight trajectory data is then outputted as a single CSV file per flight by Apache Spark in parallel. By adopting Apache Spark to process the raw SFDPS messages, the required time to perform all the aforementioned operations on SFDPS messages spanning a time duration of 24 h decreases dramatically from nearly one week to approximately 10 h on a Windows 10 desktop with an Inter Core i7-4790 CPU (3.60 GHz) and 16 GB memory. Table 1 shows a small portion of processed tracking information for an American Airlines flight with Call Sign (or flight ID) AAL10 that departed from Los Angeles International Airport (ICAO code: KLAX) and arrived at John F. Kennedy International Airport (ICAO code: KJFK) on January 11, 2019.

Significant flight trajectory information, such as timestamp, flight position, and flight state derived from the SFDPS messages, is illustrated in Table 1. To be more specific, the second column indicates the airline that operates this flight, and the next two columns show the departure and arrival airports as represented by four-letter ICAO airport codes, respectively. The next two columns (position time and target position time) represent the time instant associated with the current position of an active flight and the time associated with the raw radar return, respectively. As can be seen from the column "position time", flight position comprising of latitude, longitude, and altitude is periodically updated approximately at intervals of 12 s, except the shaded cells. The next three columns show the aircraft state at the corresponding position time. Specifically, the column "actual speed" denotes the ground speed of the flight (unit: knots per hour). The next two columns describe the aircraft speed along the X and Y components tracked by radar surveillance. Similarly, the column "target altitude" represents the assigned altitude or flight level in feet. For example, an assigned altitude of 1700 means that the aircraft is to fly at 1700 ft. The column "actual altitude" denotes the actual flight level at each time instant. The column "actual position" indicates the actual location of an active flight in latitude/longitude format as reported by surveillance, and the column "target position" shows the target position of the flight at a given time instant.

As implied by the variation of actual flight altitude shown in Table 1, we observe that taking off and landing are also included in the derived flight trajectory. With respect to the flight speed, it gradually increases from 210 knots per hour to the maximum value 532 knots per hour during the cruise phase. During the landing phase, the actual flight speed significantly reduces in accordance with the decrease of altitude until touchdown. With the derived flight trajectory, we project the trajectory data on the map to check whether any large segment of flight trajectory is missing. Fig. 5 shows the visualization of the full trajectory of two sample flights, namely American Airline AAL10 from Los Angeles International Airport (LAX) to John F. Kennedy International Airport (JFK); and Delta Airline DAL2775 from Seattle Tacoma International Airport (SEA) to Hartsfield Jackson Atlanta International Airport (ATL) both on January 11, 2019. If the streamed data is incomplete, the missing data should be reflected in most flight trajectories. The visualization of sample flight trajectories facilitates fast visual inspection of flight trajectories, and it helps us quickly verify whether the data streamed from FAA is intact or not. After the visual inspection, we develop computer programs to perform strict quantitative verification on the integrity of the streamed data.

4.2. Model construction

In this section, two probabilistic deep learning models are trained to predict flight trajectory from different perspectives. The probabilistic DNN models are trained with historical data to forecast the deviation between filed flight plan and actual flight trajectory; while the probabilistic LSTM models are constructed to make longer-term prediction on the state of flight trajectory.

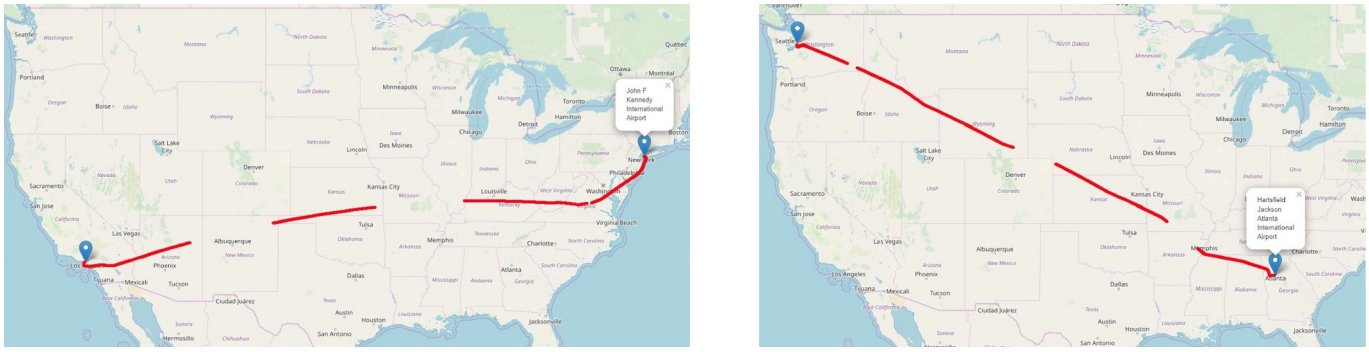
4.2.1. Data processing

Prior to model construction, historical flight trajectory data needs to be preprocessed such that the reported flight positions have equal time intervals. In this regard, two specific issues need to be addressed. First of all, since an aircraft can be tracked by several radars simultaneously, the same flight position and state might be reported multiple times, as shown by the green shaded cells in Table 1. Besides, the time interval between two consecutively reported flight positions might be much less than the regular position tracking frequency of 12 s. For example, the duration between two timestamps shown in the 20th and 21st rows is only 7 s, as shown by the blue shaded cells in Table 1. The rows with duplicated records can be dropped in a straightforward manner by retaining only one unique record in the dataset. For the case of excessive records as indicated in the blue shaded cells in Table 1, we develop a method to maintain the correct records subject to the 12 s time interval constraint. As shown in Fig. 6, suppose the timestamp of the current record is '2019-01-11T**04:08:47**.000-06:00', then the expected timestamp of the next record should be '2019-01-11T**04:08:59**.000-06:00'. Next, we define a search region comprised of a fixed number of four rows following the current record, and identify the record (within the search region) that has timestamp closest to the expected timestamp of the next record, which is '2019-01-11T04:08:58.000-06:00' in Table 1. All the records between the current record and the identified next record are then eliminated accordingly. Afterwards, we treat the identified next record as the current record and start searching for the new next record within the next four rows. The same procedures are repeated until all the excessive records are eliminated from the dataset.

On the other hand, the red shaded cells in Table 1 indicate that the time interval between the two records is 24 s(other values like 36 s, 48 s etc. are also observed in the original dataset), which implies that one or more records are missing. To address this issue, we first determine how many records are missing between two available records, then use linear interpolation to make up the missing records for the sake of simplicity. With the aforementioned operations, we ensure that all the flight positions in the dataset have approximately equal time intervals. The equal time intervals play a significant role in the subsequent model construction.

Table 1
Sample flight trajectory parsed from SFDPS messages.

ID	Airline	Arrival Point	Departure Point	Position Time	Target Position Time	Actual Speed	X velocity	Y velocity	Actual Altitude	Target Altitude	Actual Position	Target Position
1	AAL	KJFK	KLAX	2019-01-11T23:16:26.000-06:00	2019-01-12T05:16:26Z	210	-201	-61	2100	2100	33.930556 -118.457778	33.930556 -118.457778
2	AAL	KJFK	KLAX	2019-01-11T23:16:38.000-06:00	2019-01-12T05:16:38Z	210	-190	-89	2300	2300	33.924722 -118.470278	33.924722 -118.470000
3	AAL	KJFK	KLAX	2019-01-11T23:17:02.000-06:00	2019-01-12T05:17:02Z	218	-171	-135	2600	2600	33.908333 -118.492778	33.908333 -118.493333
4	AAL	KJFK	KLAX	2019-01-11T23:17:14.000-06:00	2019-01-12T05:17:14Z	238	-145	-188	3000	3000	33.896667 -118.503611	33.897778 -118.504167
5	AAL	KJFK	KLAX	2019-01-11T23:17:25.000-06:00	2019-01-12T05:17:25Z	248	-89	-232	3600	3600	33.883889 -118.508611	33.884722 -118.509167
6	AAL	KJFK	KLAX	2019-01-11T23:17:37.000-06:00	2019-01-12T05:17:37Z	246	-20	-245	4200	4200	33.870278 -118.509167	33.871389 -118.510278
7	AAL	KJFK	KLAX	2019-01-11T23:17:49.000-06:00	2019-01-12T05:17:49Z	242	-12	-242	4800	4800	33.857222 -118.510000	33.857500 -118.510556
8	AAL	KJFK	KLAX	2019-01-11T23:18:01.000-06:00	2019-01-12T05:18:01Z	242	-12	-241	5400	5400	33.843889 -118.510833	33.844722 -118.512222
9	AAL	KJFK	KLAX	2019-01-11T23:18:13.000-06:00	2019-01-12T05:18:13Z	242	-12	-241	5800	5800	33.830556 -118.511667	33.831389 -118.513333
10	AAL	KJFK	KLAX	2019-01-11T23:18:25.000-06:00	2019-01-12T05:18:25Z	242	-13	-241	6300	6300	33.816667 -118.512778	33.817222 -118.513611
11	AAL	KJFK	KLAX	2019-01-11T23:18:37.000-06:00	2019-01-12T05:18:37Z	243	-13	-242	6800	6800	33.803056 -118.513611	33.803333 -118.515000
12	AAL	KJFK	KLAX	...	...	...	...	...	...	...	...	...
13	AAL	KJFK	KLAX	2019-01-11T02:30:22.000-06:00	2019-01-11T08:30:22Z	528	528	16	35000	35000	37.926667 -87.807222	37.928056 -87.808056
14	AAL	KJFK	KLAX	2019-01-11T02:30:22.000-06:00	2019-01-11T08:30:22Z	532	532	15	35000	35000	37.928333 -87.806667	37.928889 -87.807222
15	AAL	KJFK	KLAX	2019-01-11T02:30:34.000-06:00	2019-01-11T08:30:34Z	528	528	16	35000	35000	37.927778 -87.770000	37.929444 -87.770556
16	AAL	KJFK	KLAX	2019-01-11T02:30:34.000-06:00	2019-01-11T08:30:34Z	532	532	15	35000	35000	37.929444 -87.769722	37.930278 -87.769722
17	AAL	KJFK	KLAX	2019-01-11T02:30:46.000-06:00	2019-01-11T08:30:46Z	528	528	17	35000	35000	37.928611 -87.733056	37.928889 -87.734167
18	AAL	KJFK	KLAX	2019-01-11T02:30:46.000-06:00	2019-01-11T08:30:46Z	532	531	15	35000	35000	37.930000 -87.732778	37.929722 -87.733333
19	AAL	KJFK	KLAX	...	...	...	...	...	...	...	...	...
20	AAL	KJFK	KLAX	2019-01-11T04:08:47.000-06:00	2019-01-11T10:08:47Z	126	-108	64	1300	1300	40.613611 -73.695278	40.613056 -73.695000
21	AAL	KJFK	KLAX	2019-01-11T04:08:54.000-06:00	2019-01-11T10:08:54Z	139	-116	76	1300	1300	40.617500 -73.700556	40.616667 -73.700278
22	AAL	KJFK	KLAX	2019-01-11T04:08:57.000-06:00	2019-01-11T10:08:57Z	126	-109	63	1200	1200	40.616667 -73.701389	40.617222 -73.700833
23	AAL	KJFK	KLAX	2019-01-11T04:09:11.000-06:00	2019-01-11T10:09:11Z	126	-109	63	1100	1100	40.620556 -73.710000	40.621111 -73.708056
24	AAL	KJFK	KLAX	2019-01-11T04:09:24.000-06:00	2019-01-11T10:09:24Z	125	-108	62	900	900	40.623889 -73.718611	40.623333 -73.718056
25	AAL	KJFK	KLAX	2019-01-11T04:09:34.000-06:00	2019-01-11T10:09:34Z	122	-105	62	800	800	40.626944 -73.723889	40.627500 -73.720833
26	AAL	KJFK	KLAX	2019-01-11T04:09:48.000-06:00	2019-01-11T10:09:48Z	122	-104	63	600	600	40.631111 -73.732222	40.631667 -73.729722
27	AAL	KJFK	KLAX	2019-01-11T04:10:02.000-06:00	2019-01-11T10:10:02Z	122	-104	64	500	500	40.634722 -73.740833	40.634167 -73.741111
28	AAL	KJFK	KLAX	2019-01-11T04:10:11.000-06:00	2019-01-11T10:10:11Z	121	-103	63	400	400	40.636944 -73.746389	40.636389 -73.747222
29	AAL	KJFK	KLAX	2019-01-11T04:10:15.000-06:00	2019-01-11T10:10:15Z	121	-103	63	300	300	40.638333 -73.749167	40.638611 -73.748611
30	AAL	KJFK	KLAX	2019-01-11T04:10:31.000-06:00	2019-01-11T10:10:15Z	121	-103	63	300	300	40.643056 -73.758889	40.638611 -73.748611



(a) American Airlines AAL10: Los Angeles International Airport (LAX) to John F. Kennedy International Airport (JFK)

(b) Delta Airlines DAL2775: Seattle Tacoma International Airport (SEA) to Hartsfield Jackson Atlanta International Airport (ATL)

Fig. 5. Visualization of two sample flight trajectories: AAL10 and DAL2775 on January 11, 2019.

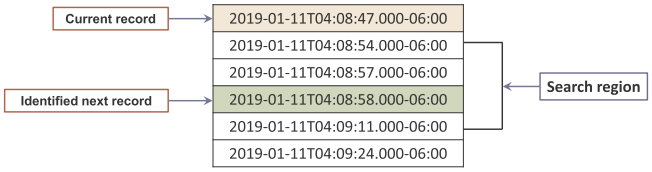


Fig. 6. Illustration of developed method for filtering out excessive records.

4.2.2. *Trajectory deviation prediction by feedforward deep neural network*

Considering the information shown in Table 1, flight trajectory prediction can be carried out from several different perspectives. Since the flight plan is already given, one straightforward approach is to forecast the deviation between the actual flight trajectory and the filed target trajectory over time. Considering it is fast and efficient to measure model prediction uncertainty with MC dropout in deep neural networks, we proceed with the development of DNN models to predict the trajectory deviation in the next time instant, with the current deviation calculated based on the filed flight plan and the current aircraft state. To accomplish this goal, we build feedforward neural networks consisting of three hidden layers with each hidden layer having 32, 64, and 32 hidden units, respectively, and use rectified linear units (ReLU) as the activation functions across all the layers. Since the DNN model uses flight state and current trajectory deviation as input variables, it can only make a one-step-ahead prediction because we could only observe actual flight trajectory and flight state up to the current time instant. Fig. 7 shows the distribution of trajectory deviation of a sample flight with flight ID AAL598. As can be observed, flight trajectory deviation primarily occurs along the dimension of latitude and longitude, while the altitude of the actual trajectory complies with the target altitude quite well across the entire course of the flight. Such phenomenon is observed in almost all of the commercial flights we analyzed. From this standpoint, since flight altitude almost never deviates from the target altitude, we only need to build two DNN models of the same structure to predict the trajectory deviation along the latitude and longitude.

In the deep feedforward neural networks for flight trajectory deviation prediction, we regard the latest flight state and trajectory deviation at the current time instant t as model inputs, and the model output is the deviation from the target trajectory in the next time instant $t + 1$. Fig. 8 illustrates the framework of the constructed deep feedforward neural networks. As can be seen, there are five input variables, namely trajectory deviation at the current time instant t, aircraft velocity along X and Y, current altitude and aircraft speed. Our goal is to predict trajectory deviation at the next time instant $t + 1$. As mentioned earlier, one important consideration is to account for model prediction uncertainty (e.g., parameter uncertainty and model structure uncertainty) given so many DNN parameters to be estimated in the model. To do this, we perform MC dropout with 10% of units randomly dropped before each hidden layer to approximate the Bayesian posterior distribution, from which we obtain a predictive distribution for the latitude and longitude deviation for the next time instant.

By doing this, we forecast the trajectory deviation along latitude and longitude for the next time instant $t + 1$. The trajectory deviation prediction enables us to estimate the future position of an ongoing flight with a high accuracy.

4.2.3. *Flight state prediction by LSTM*

Since massive historical flight trajectories are readily available, another approach to trajectory prediction problem is to mine complex

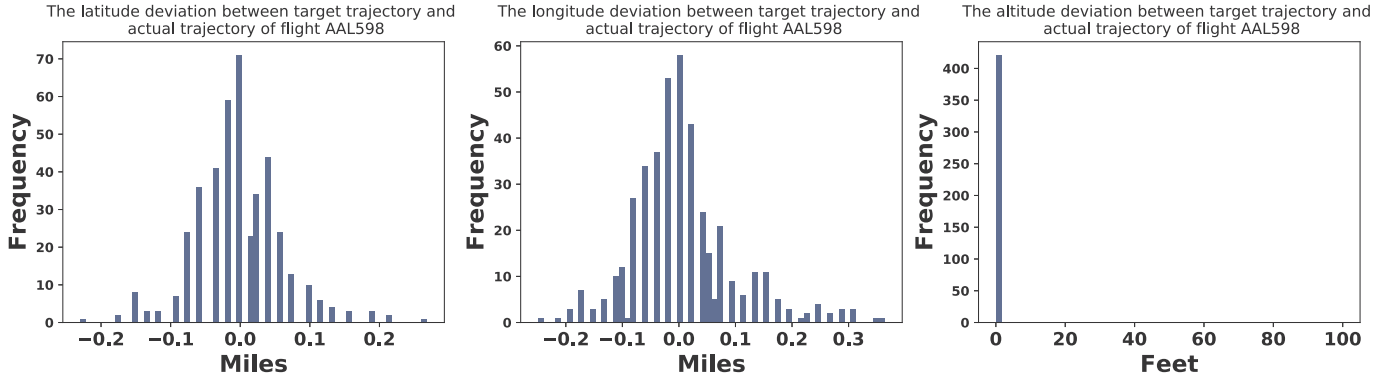


Fig. 7. Trajectory deviation of a sample flight with call sign AAL598, from New York LaGuardia Airport (LGA) to Charlotte Douglas International Airport (CLT).

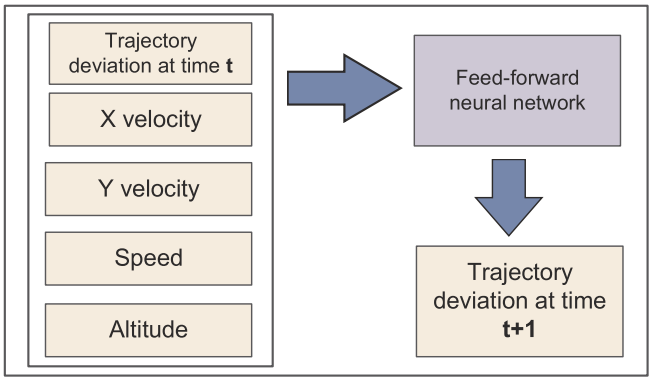


Fig. 8. Framework of the feedforward deep neural network.

flight patterns from the historical data; then the patterns learned from the historical data can be further utilized to predict the trend of an ongoing flight with the same origin and destination. In this regard, recurrent neural networks (RNNs) have a powerful learning capability and have demonstrated remarkable effectiveness in various domains, such as image recognition [44], speech recognition [45, 46], machine translation [47], and others [48-50].

In this paper, we work along this direction and train two LSTM models of the same structure (based on the categories mentioned below) to make prediction on the full state of flight trajectory. The detailed architecture of the LSTM model is shown in Fig. 9. As can be observed, two hidden LSTM layers are stacked together, in which each hidden layer consists of 20 LSTM blocks and each LSTM block has 50 dimensions. Every time, 20 time steps of past flight trajectory $(\boldsymbol{x}_{t-19}, \cdots, \boldsymbol{x}_t)$ comprised of latitude, longitude, altitude, flight velocity along X and Y, and speed, are fed into the constructed LSTM model, and the output of the last LSTM block at the top hidden layer is fed into a conventional feedforward densely connected layer, which maps the intermediate LSTM output to the flight trajectory in the subsequent five time instants $(\hat{\boldsymbol{x}}_{t+1}, \cdots, \hat{\boldsymbol{x}}_{t+5})$. The predicted flight trajectory $(\hat{\boldsymbol{x}}_{t+1}, \cdots, \hat{\boldsymbol{x}}_{t+5})$ can be further used as new inputs to the LSTM model, then the trained model is used to make prediction on the future flight trajectory again $(\hat{\boldsymbol{x}}_{t+6}, \cdots, \hat{\boldsymbol{x}}_{t+10})$ as shown in the blue dashed boxes in Fig. 9. The bold red and green arrows in Fig. 9 indicate the place where dropout occurs.

In the constructed LSTM model, since we want to predict the state of the flight trajectory, the number of the input features is the same as that of output variables. Since there are six variables in total, we group them into two categories: latitude and longitude, altitude and flight velocity along X and Y (since flight speed can be calculated from flight velocity along X and Y easily, thus it is ignored here). If we only build one model where all the variables are used as outputs, then the model will be driven by the variable with the largest loss value in each iteration. Under this circumstance, the weights of the LSTM model will be optimized along the direction that minimizes the loss function of the variable with the largest loss. Since the five response variables share many common parameters in the first two LSTM layers, the dominance of the variable with the largest loss results in the ignorance of the optimization of the weights for the remaining variables. Therefore, we categorize the five response variables with similar loss values in the same

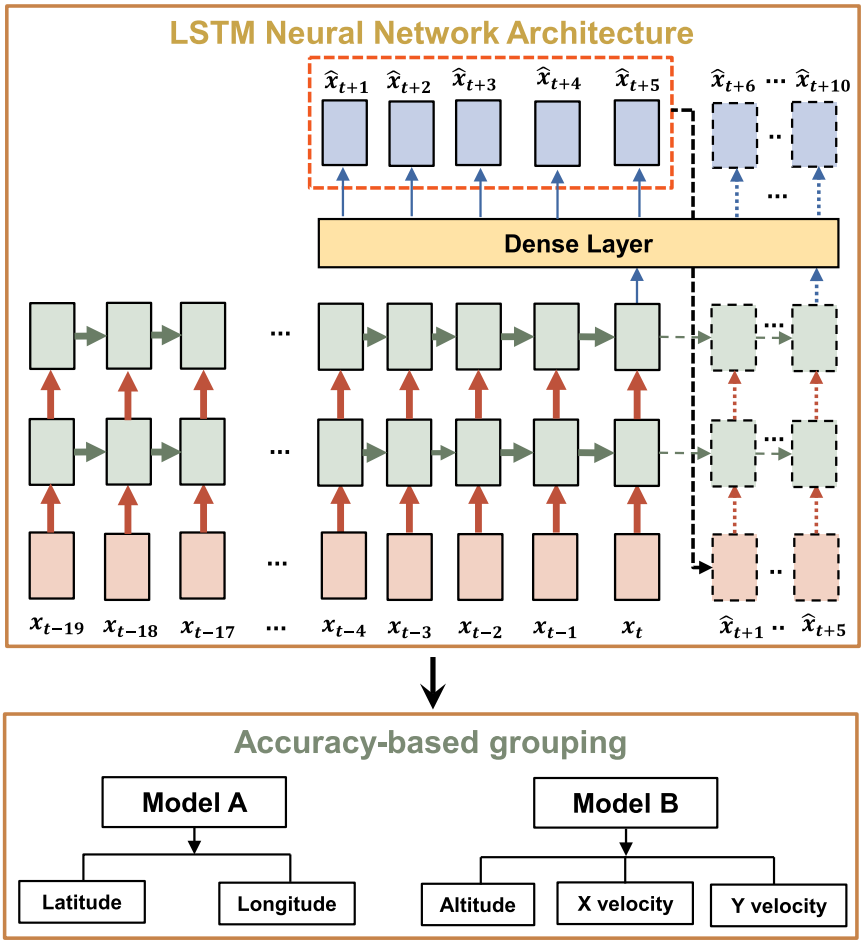


Fig. 9. Architecture of the proposed long short-term memory (LSTM) neural network. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

group, and train two individual LSTM models to overcome the deficiency in building one single model (see model A and model B in Fig. 9). By grouping the variables with similar loss values together, we also maintain the different prediction precision demand specific to each variable. In particular, since a small prediction error in latitude and longitude is a large value in terms of distance in reality, there is an important need for high prediction precision for both variables. If the other variables (e.g., velocity) dominate the loss function, then the prediction precision for latitude and longitude will be compromised significantly.

To obtain the Bayesian posterior distribution for model prediction, we perform Monte Carlo dropout for both inputs, outputs, and recurrent layers with a dropout probability of 0.1 following the guidance in Ref. [51]. The MC dropout enables us to have an approximated Bayesian posterior distribution on the future flight trajectory of interest.

4.3. Model integration

As introduced before, two different types of deep learning models are trained to make predictions on flight trajectory from different perspectives. Specifically, the feedforward deep neural networks (DNN) make a one-step-ahead prediction with high accuracy on the deviation of the actual flight trajectory away from the filed flight plan, while the LSTM models forecast the full state of flight trajectory for a longer term (i.e., five steps or longer) but with a lower accuracy in comparison with the DNN models. The integration of the two types of models could have both high prediction accuracy and longer-term prediction capability.

Suppose we have observations on the actual flight trajectory up to the time instant t; the DNNs predict flight trajectory deviation at the next time instant $t+1$ based on the degree of trajectory deviation at the previous time instant t. Given the predicted deviation at time instant $t+1$, the predicted flight position at time $t+1$ by the DNNs is readily available by combining the predicted deviation and the filed flight plan. Suppose the LSTM models utilize the past 20 observed trajectory data to predict the state of flight along five subsequent time instants $t+1,\cdots,t+5$. Let DNNs' trajectory deviation prediction at time instant $t+1$ be denoted by $\boldsymbol{\tau}_{t+1}$; then we have:

$$E(\hat{\boldsymbol{y}}_{t+1}) = \boldsymbol{q}_{t+1} + E(\boldsymbol{\tau}_{t+1}) \tag{4}$$

where $\boldsymbol{q}_{t+1}$ denotes the planed flight position including latitude, longitude, and altitude at the time instant $t+1$, $E(\boldsymbol{\tau}_{t+1})$ is the mean value of the DNN trajectory deviation prediction at time instant $t+1$, and $E(\hat{\boldsymbol{y}}_{t+1})$ represents the mean value of the DNN prediction of the flight position at time instant $t+1$.

Considering the two response variables (latitude and longitude) shared by the two models, one straightforward way to integrate the two models is to correct the low-accuracy model with the result from high-accuracy model through a discrepancy term as calculated following Eq. (5).

$$\boldsymbol{\Delta}_{t+1} = E(\hat{\boldsymbol{y}}_{t+1}) - E(\hat{\boldsymbol{x}}^{c}_{t+1}) \tag{5}$$

where $E(\hat{\boldsymbol{x}}^{c}_{t+1})$ denotes the mean value of LSTM model's prediction of the two common quantities (latitude and longitude) at the time instant $t+1$, and $\boldsymbol{\Delta}_{t+1}$ represents the discrepancy between DNN's mean predictions and LSTM model's mean predictions.

Once the discrepancy between the two model predictions of latitude and longitude at time $t+1$ is estimated, then the LSTM model's prediction of the two quantities for the subsequent time instants $t+1,\cdots,t+5$ can be updated accordingly by adding the discrepancy term to its predictions as shown in Eq. (6).

$$\bar{\boldsymbol{x}}^{c}_{i} = \hat{\boldsymbol{x}}^{c}_{i} + \boldsymbol{\Delta}_{t+1}, \quad \forall\, i \in [t+1,\cdots,t+5] \tag{6}$$

where $\bar{\boldsymbol{x}}^{c}_{i}$ denotes the updated LSTM model prediction on latitude and longitude for the subsequent time instants $t+1,\cdots,t+5$.

Another way to integrate the two models is to let the DNN model make only one-step correction at the time instant $t+1$. However, since the correction $\boldsymbol{\Delta}_{t+1}$ at time step $t+1$ is quite small compared to $\hat{\boldsymbol{x}}^{c}_{t+1}$, considering that LSTM takes the data over the past 20 time instants as inputs, the effect of one-step correction on LSTM's prediction will be negligible. The effect will further weaken with the increase of the LSTM prediction step. Hence, we integrate the two models by letting the DNN make corrections on all subsequent $t+1,\cdots,t+5$ time instants at the same time in consideration of the high prediction accuracy of the DNN model. By combining the DNN prediction $\hat{\boldsymbol{y}}_{t+1}$ with the LSTM model prediction $\hat{\boldsymbol{x}}_{t+1}$, both high accuracy and longer-term prediction capability are achieved in the integrated model.

4.4. Safety measure

The objective of trajectory prediction is to assess the en-route flight safety of the NAS. Separation distance, as a widely used safety metric during the en-route phase, is used as a quantitative metric in this paper to measure the system safety. In the en-route airspace, the minimum horizontal separation distance is 5 nautical miles, while no aircraft should come vertically closer than 300 m at an altitude of 29,000 ft. [11]. With the probabilistic predictions of the trajectory of any two given flights, the safety metric used in this paper is mathematically described in the following equation.

$$I\left[p(\boldsymbol{d}^{h}_{i}(\bar{\boldsymbol{x}}^{c}_{i}(A), \bar{\boldsymbol{x}}^{c}_{i}(B)) < \delta^{h}) > \lambda \textbf{ and } p(\boldsymbol{d}^{v}_{i}(\bar{\boldsymbol{x}}^{c}_{i}(A), \bar{\boldsymbol{x}}^{c}_{i}(B)) < \delta^{v}) > \lambda\right] \quad \forall\, i \in [t+1,\cdots,t+5] \tag{7}$$

where $\bar{\boldsymbol{x}}^{c}_{i}(A)$ and $\bar{\boldsymbol{x}}^{c}_{i}(B)$ denote the integrated model's predictions of the trajectories of flights A and B, respectively; $\boldsymbol{d}^{h}_{i}(\bar{\boldsymbol{x}}^{c}_{i}(A), \bar{\boldsymbol{x}}^{c}_{i}(B))$ and $\boldsymbol{d}^{v}_{i}(\bar{\boldsymbol{x}}^{c}_{i}(A), \bar{\boldsymbol{x}}^{c}_{i}(B))$ represent the horizontal and vertical separation distance between the two flights A and B as forecast by the integrated model. Since the trajectory prediction of the integrated model is a probabilistic quantity, a probabilistic metric $p(\boldsymbol{d}^{h}_{i}(\bar{\boldsymbol{x}}^{c}_{i}(A), \bar{\boldsymbol{x}}^{c}_{i}(B)) < \delta^{h})$ and $p(\boldsymbol{d}^{v}_{i}(\bar{\boldsymbol{x}}^{c}_{i}(A), \bar{\boldsymbol{x}}^{c}_{i}(B)) < \delta^{v})$ are defined to measure the degree to which the horizontal and vertical separation distance are violated against the recommended threshold values δ^{h} and δ^{v}, respectively. Herein, $I[\cdot]$ is an indicator function; if both the horizontal and vertical separation distances are violated, then the indicator function takes a value of one, indicating that the safety of the two flights in terms of separation distance is compromised. Otherwise, its takes a value of zero to indicate that the two flights are safe in terms of separation distance.

With the introduction of the probabilistic safety indicator, the trajectory prediction from the integrated model is related to the safety condition between any two given flights. In particular, since the LSTM model has a longer-term prediction, if the probability of separation distance violation forecast by the integrated model is above a threshold probability value, then the ground controllers and flight operators could be alerted to take appropriate actions to increase the separation distance between the two flights.

4.5. Summary

The methodology developed in this section tackles the problem of en-route flight trajectory prediction in the NAS following a four-step procedure (see Fig. 4).

1. Considering the large volume of raw SFDPS messages in FIXM format, Apache Spark is leveraged to extract important trajectory information (i.e., flight position, aircraft velocity) pertaining to each flight. SQL query is performed to group flight trajectory by flight ID and date, and to save flight trajectory as a single CSV file per flight with Apache Spark. Data pre-processing techniques are used to drop redundant data and impute missing data.
2. Two different types of deep learning models are trained to predict flight trajectory from different angles: deep feedforward neural

networks are constructed for one-step-ahead prediction of the deviation of actual flight trajectory from its corresponding flight plan, and LSTM neural networks are developed to make longer-term prediction of the flight trajectory. The uncertainty in both models is quantified following a Bayesian approach and approximated with Monte Carlo dropout.

3. The two deep learning models are integrated through a discrepancy term calculated as the difference between the two models' mean predictions, and correcting the LSTM model prediction using the discrepancy term. By doing this, the corrected model retains both LSTM models' longer-term prediction capability and DNN models' higher prediction accuracy.
4. Using the integrated model, trajectory prediction is executed for multiple adjacent flights in a probabilistic manner. A probabilistic separation distance-based safety metric is used to measure en-route flight safety in a quantitative manner. The real-time en-route safety assessment helps to monitor the system safety and prevent the occurrence of safety hazard in advance.

5. Computational results

In this section, we illustrate the computational results of the developed deep learning models, and demonstrate the superior performance of the integrated model over the two individual deep learning models in trajectory prediction.

5.1. Data

We streamed SFDPS messages from 19th December 2018 to 8th February 2019. In terms of data volume, nearly 4.2 TB of SFDPS messages is processed with Apache Spark, from which we extract flight trajectories for 48 days (FAA server was down for a few days). Since the deep learning model is specific to each flight ID, we identify 3789 flights that are present every day from 19th December 2018 to 8th February 2019. Theoretically, we can build 3789 deep learning models with each model per flight ID. In order to demonstrate the prediction performance of the deep learning models, we pick a specific flight AAL598 for the sake of illustration. Since the objective of the deep learning models (flight ID-wise deep learning model) is to predict the trajectory of an ongoing flight with the same flight ID, thus we perform leave-one-out cross validation for the trajectories of flight AAL598 over the past 48 days, and the result is reported in Section 5.3. The source codes for the developed models are available at GitHub via the link https://github.com/zxgcqupt/SWIM_Public.

Fig. 10 shows the historical trajectories of flight AAL598 from 19th December 2018 to 8th February 2019. The bottom left and top right corners indicate the departure airport (LaGuardia Airport) and destination airport (Charlotte Douglas International Airport), respectively. Obviously, the flight trajectory varies from day to day as impacted by traffic flow control, different weather conditions, and other factors.

5.2. Model training and prediction analysis

Prior to the training of the DNN model, the input and output data are normalized with the MinMaxScaler. The hyperparameters (e.g., learning rate, dropout rate, and iterations) of the DNN model are tuned with grid search. To be specific, grid search is used to determine the value of learning rate, which increases from 0.0001 to 0.1 ([0.0001, 0.001, 0.005, 0.008, 0.01, 0.05, 0.08, 0.1]), the iterations to train the deep learning models vary from 1000 to 20000 ([1000, 2000, 4000, 6000, 8000, 10,000, 12,000, 14,000, 16,000, 18000, 20,000]), and the dropout rate varies over within the range [0.02, 0.2] ([0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20]). The grid search shows that the DNN model using a dropout rate of 0.1 with 8000 iterations, and Adam optimizer with a learning rate of 0.001, results in the model having minimum prediction error. Two DNN models are trained to predict the trajectory deviation along latitude and longitude, respectively. When making predictions, Monte Carlo dropout with the same probability is used to approximate the distribution of deviation at the next time instant. Fig. 11 shows the trajectory deviation prediction by the DNN model on the test flight AAL598. As can be observed from Fig. 11, the trained DNN model captures most of the variation on the trajectory deviation over time.

To compare the computational results of the deterministic and probabilistic models, Fig. 12 illustrates the one-step-ahead prediction of the latitude and longitude in the trajectory of flight AAL598. With the trajectory deviation forecast by the DNN models, the trajectory in terms of latitude and longitude in the next time instant can be calculated in a straightforward manner. As shown in Fig. 12, the actual flight position is denoted as solid green dots, the prediction of deterministic model on flight trajectory is represented as solid yellow dots, the solid pink dots are 1000 samples drawn from the trained DNN model, and the black crosses denote the mean prediction of the probabilistic model. In terms of computational time, it takes the trained model 0.11 s on average to draw 1000 samples when making predictions on one data point, which is significantly less than the rate at which flight position gets updated (12 s per position update). As can be observed, the samples from the trained deep Bayesian neural networks form a pink cluster, and the dispersion of the cluster represents the uncertainty in predicting the flight trajectory. The inclusion of model prediction uncertainty enables

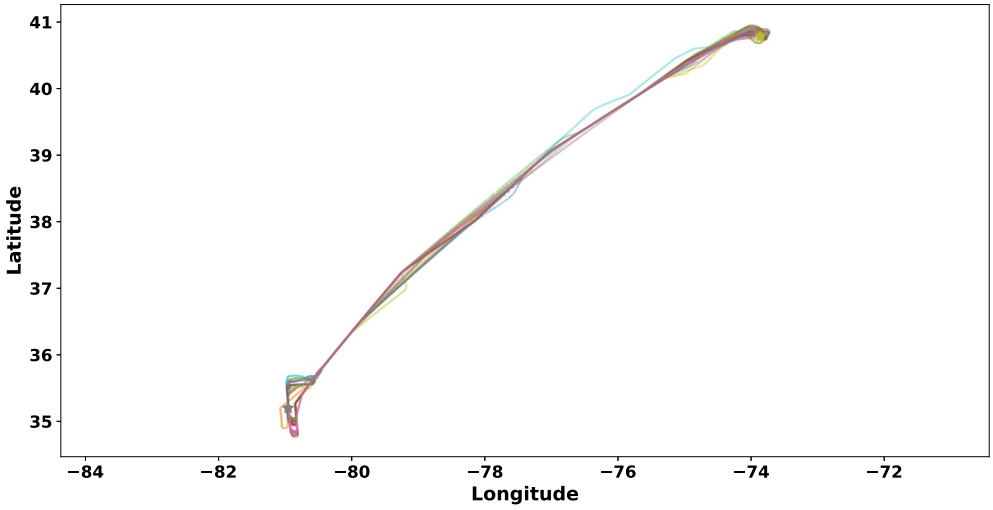
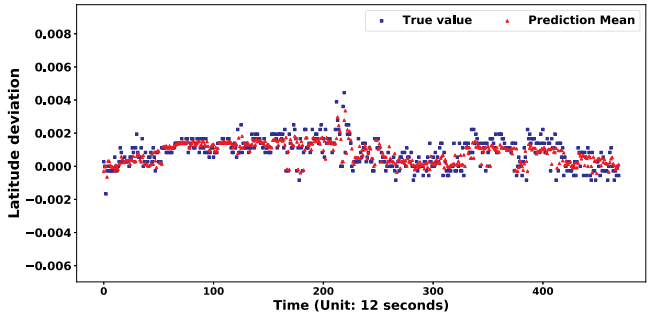
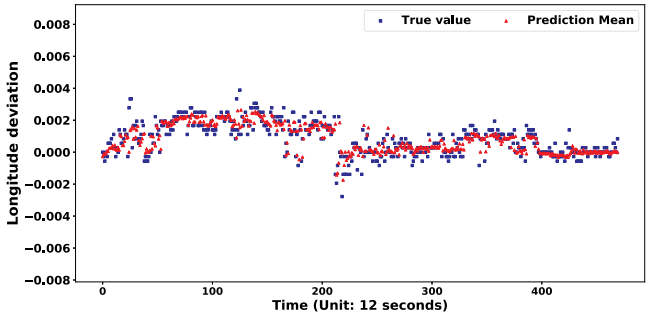


Fig. 10. The historical trajectories of flight AAL598 from 19th December 2018 to 8th February 2019.



(a) Mean latitude deviation prediction vs actual values on sample flight AAL598



(b) Mean longitude deviation prediction vs actual values on sample flight AAL598

Fig. 11. DNN prediction on trajectory deviation.

us to evaluate the confidence in the predicted value and quantitatively assess the level of risk involved when making decisions.

We retain the same trajectory to verify the performance of the trained LSTM model. To avoid model overfitting, regularization is performed on layer parameters, and $L2$ regularization is applied to the recurrent connections. The LSTM model has two stacked layers, and the prefix of LSTM neural network is tuned. When the LSTM prefix is 20, the prediction error is minimized. The output of the last LSTM block at the top hidden layer is flattened and fed into a densely connected layer. Two individual models are trained with the same LSTM architecture. The first model predicts the flight latitude and longitude along the subsequent five time steps, while the second model forecasts the flight altitude, and velocities along X and Y in the subsequent five time steps. To make even longer prediction, we take the LSTM model's predictions as new inputs, and feed them into the trained LSTM model to make predictions again. Fig. 13 shows the predictions of the LSTM model of the latitude, longitude, and altitude of the test flight AAL598 for the next 10 time instants (2 min). From the LSTM predictions, we observe that it demonstrates promising performance in predicting the flight trajectory over time. There is a slight discrepancy between the LSTM model predictions and the actual values. The powerful longer-term prediction ability in the LSTM model enables us to have a 2-minutes-ahead view of the possible behavior of each flight. As indicated in Ref. [52], the median response time (i.e., the time after the alert activated until the controller begins to issue a control instruction) following a Conflict Alerts (CAs) is 88 s, and the median response time following an Minimum Safe Altitude Warnings (MSAWs) is 38 s. From this perspective, the 2-minutes-ahead view offers enough time for air traffic controllers to assess the situation, and take actions to mitigate potentially hazardous events.

Both the DNN and LSTM models make predictions on the same flight trajectory from different perspectives. By combining the two predictions, both high accuracy and longer-term prediction ability are achieved in the integrated model, thus supporting more robust decision making. As indicated in Fig. 7, since the deviation along latitude and longitude is small for DNN, the DNN prediction can be utilized to correct the LSTM model's prediction on the common quantities of the two models. Fig. 14 compares the predictions of the LSTM model before and after the integration with the DNN model. It can be observed that the integrated model makes a much more accurate prediction on the flight trajectory than the predictions from the LSTM model only. The prediction performance of the integrated model deteriorates as the number of look-ahead steps increases. Specifically, the prediction error gradually increases to 1.75 miles (0.025×70) in the 10-th step-ahead prediction.

To measure the flight safety, we pick another flight UAL1767 that starts from Denver International Airport and ends at Ronald Reagan Washington National Airport on 23rd January, 2019. Once the deep learning models are trained, we calculate the horizontal and vertical separation distances between the two flights. Fig. 15 compares the model prediction and the ground truth of the separation distance between the two flights. In this figure, the red dashed lines indicate the actual horizontal and vertical separation distances between the two flights, the black solid lines denote the mean value of separation distance as estimated by the probabilistic models, and the shaded areas in dark grey and light grey demonstrate the confidence interval for one standard deviation and two standard deviations away from the mean value, respectively. Table 2 shows the MAE of spatial distance and the

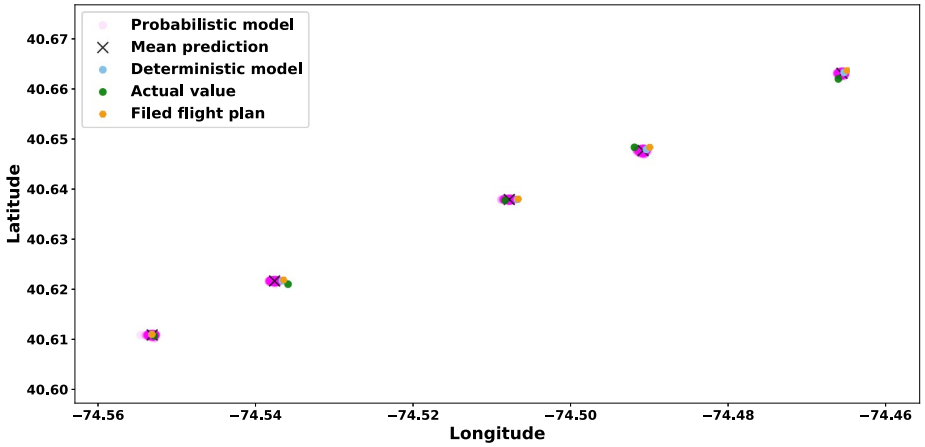
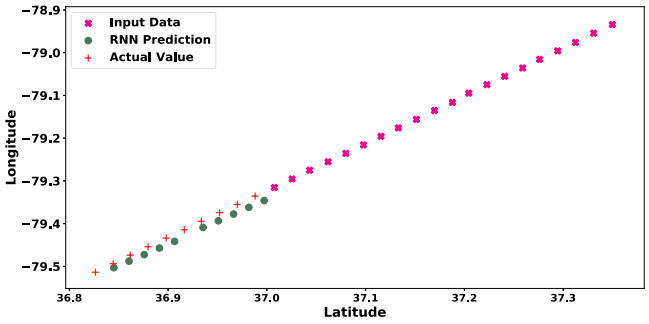
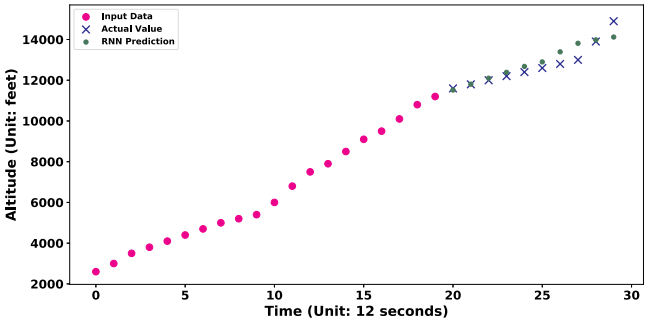


Fig. 12. Comparisons of the trajectory predictions of flight AAL598. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)



(a) Mean latitude and longitude predictions by the LSTM model vs actual values on sample flight AAL598



(b) Mean altitude prediction by the LSTM model vs actual values on sample flight AAL598

Fig. 13. LSTM prediction of flight trajectory.

proportion between prediction error and the actual values. As can be observed, the deep learning model captures both separation distances very well. The probabilistic metric formulated in Eq. (7) indicates that two flights did not violate the separation distance constraint.

5.3. Model performance cross validation

absolute error (MAE) between the model predictions and actual values is used as a metric to compare all the models. Algorithm 1 summarizes the procedures of leave-one-out cross-validation, where y_i denotes the value of the i-th trajectory, and $\hat{y}^{(-i)}$ is the prediction of the model trained without using the i-th trajectory on the i-th trajectory.

Algorithm 1. Procedure of leave-one-out "yank.xm"$1,0,63y"/\60\ \47\ce:display\62\/ce"cross-validation.

Algorithm 1 : Procedure of leave-one-out cross-validation

Input: n trajectories of a flight with the same ID

Process:

1: **for** $i = 1$ to n **do**
2: Train the model on all the trajectories except the i-th trajectory;
3: Compute the test error on the held out i-th trajectory $\left|\mathbf{y_i} - \hat{\mathbf{y}}^{(-i)}\right|$;
4: **end for**
5: Average the test errors: CV $= \frac{1}{n}\sum_{i=1}^{n}\left|\mathbf{y_i} - \hat{\mathbf{y}}^{(-i)}\right|$

Output: Return CV

Section 5.2 only illustrates the performance of the DNN and LSTM models on predicting one flight trajectory. In this section, we perform full leave-one-out cross-validation to verify the performance of the trained deep learning models. As the objective of deep learning models is to predict the trajectory of one ongoing flight, we retain one full flight trajectory for test purposes, while the remaining 47 trajectories are used to train the deep learning models. The same procedure is applied once for each flight trajectory in the dataset and repeated for 48 times in total for flight AAL598. The root mean squared error (RMSE) and mean

Since training deep learning models, especially LSTM models, is time-consuming, we accelerate the running speed of the program by executing it on a NVIDIA Pascal GPU. Table 3 compares the performance of the deterministic and probabilistic deep learning models on the trajectory deviation prediction. Here, we use the mean value of the probabilistic model prediction to calculate the RMSE and MAE. The last column of Table 3 indicates the spatial distance between the predicted flight trajectory and the actual flight trajectory. From the comparison results reported in Table 3, we observe that the RMSE of the

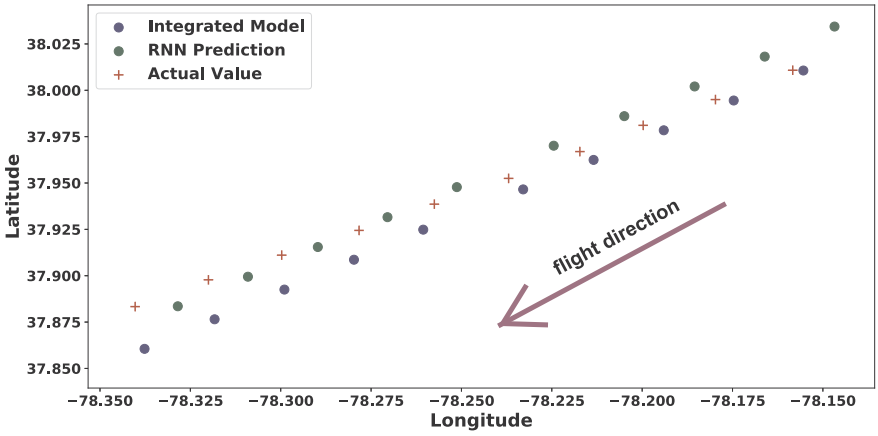


Fig. 14. Comparison of LSTM model predictions before and after integration with the DNN model.

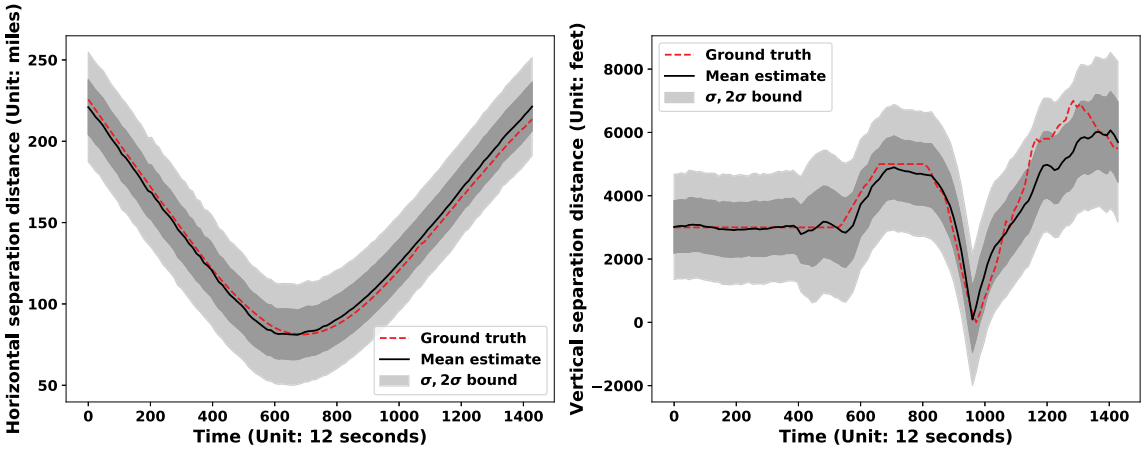


Fig. 15. Performance comparison of separation distance between AAL598 and UAL1767. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Table 2
Separation distance prediction error between flights AAL598 and UAL1767.

MAE of spatial distance (Unit: mile)	Proportion between prediction error and actual values
3.78	2.7%

probabilistic DNN model has reduced by 2.11% and 2.98% for the latitude and longitude deviation prediction, respectively. Besides, Table 3 also compares the DNN model with three baseline models: linear model, support vector machine, and decision tree regressor. Obviously, the probabilistic DNN model stands out from all the models in terms of prediction performance.

In a similar way, we compare the performance of the LSTM model and the integrated model on flight state prediction. As shown in Table 4, the integrated model demonstrates a much better performance w.r.t. the common quantities of the two models: latitude and longitude prediction. The value of RMSE is reduced by 49% and 58% compared to the LSTM model, respectively. These cross-validation results show that the integrated model achieves much higher accuracy in flight state prediction.

Next, we conduct the leave-one-out cross-validation to verify the performance of the integrated model in predicting flight trajectory for several different flights. Once we have the prediction error for a single flight, the error gets added when computing the separation distance between two flights. Following the same procedure, we build a flight ID-specific deep learning model for multiple flights, and the RMSE and MAE of model prediction is reported in Table 5. From the computational results, we observe that the integrated model has demonstrated a robust performance in predicting flight trajectory with an average prediction error of 2.4747 miles between the predicted trajectory and the actual trajectory.

6. Conclusion

In this paper, we developed and combined deep learning-based models using the SFDPS messages streamed from FAA's SWIM Flight Data Publication Service for en-route flight safety prediction. To accomplish this goal, a four-step methodology is developed. In the first step, a unified distributed high-performance computing platform Apache Spark is leveraged to parse the massive SFDPS messages in FIXM format, from which key information pertaining to flight trajectory is extracted. Next, we develop two individual deep learning models for trajectory prediction from different views. The first feedforward deep neural network is trained for one-step prediction of the deviation between actual trajectory and target flight trajectory. The second LSTM recurrent neural network is utilized to make longer-term prediction on the full state of the flight trajectory. In the third step, the two deep learning models are integrated via a discrepancy term, which is calculated as the discrepancy between DNN prediction and LSTM prediction (mean values) on the common response variables (i.e., latitude and longitude). The integration of the two deep learning models preserves both the high accuracy of the DNN model and the longer-term prediction ability of LSTM model. Finally, the same approach is extended to multiple flights, in order to predict the en-route flight safety, in which probabilistic separation distance is used as a quantitative safety metric.

The proposed methodology makes several significant contributions to the state-of-the-art towards flight trajectory prediction using machine learning from historical data. First of all, we utilize a distributed computing engine Apache Spark to process a large volume of raw data in FIXM format, and it demonstrates scalable and promising performance. The same strategy can be implemented to handle more complex aeronautical data in the future, such as weather data. Secondly, different from existing studies, we characterize the model prediction uncertainty following a Bayesian approach. The quantification of model prediction uncertainty allows us to make decisions with probabilistic

Table 3
Performance comparisons of DNN models w.r.t. trajectory prediction of flight AAL598.

	RMSE of latitude	MAE of latitude	RMSE of longitude	MAE of longitude	Spatial Distance (Unit: mile)
DNN Deterministic case	8.4675×10^{-4}	6.0573×10^{-4}	9.2802×10^{-4}	6.4842×10^{-4}	0.05929
DNN Probabilistic case	8.2883×10^{-4}	5.8234×10^{-4}	9.0035×10^{-4}	6.1728×10^{-4}	0.05678
Linear	8.6034×10^{-4}	6.1361×10^{-4}	9.5710×10^{-4}	6.6720×10^{-4}	0.06255
SVM	9.1326×10^{-4}	6.8163×10^{-4}	10.2696×10^{-4}	6.7169×10^{-4}	0.06717
DTR	10.166×10^{-4}	7.4147×10^{-4}	11.4632×10^{-4}	8.0934×10^{-4}	0.07414

[1] RMSE: root mean squared error; MAE: mean absolute error; SVM: support vector machine; DTR: decision tree regressor.
[2] Spatial distance measures the distance between the predicted latitude and longitude and the actual latitude and longitude.

Table 4
Performance comparisons of models w.r.t. the state prediction of flight AAL598.

	LSTM model A		Integrated model		LSTM model B		
	Latitude	Longitude	Latitude	Longitude	Altitude (Unit: feet)	X velocity	Y velocity
RMSE	2.93×10^{-2}	3.42×10^{-2}	1.48×10^{-2}	1.44×10^{-2}	720	17.93	18.92
MAE	2.26×10^{-2}	2.63×10^{-2}	9.61×10^{-3}	9.29×10^{-3}	530	8.33	9.08

Table 5
Performance validation of the integrated model w.r.t. flight trajectory prediction.

Flight ID	RMSE of latitude	MAE of latitude	RMSE of longitude	MAE of longitude	Spatial distance (Unit: mile)
AAL1020	5.00×10^{-2}	2.36×10^{-2}	4.02×10^{-2}	2.45×10^{-2}	2.3481
AAL1023	4.20×10^{-2}	2.37×10^{-2}	2.47×10^{-2}	1.52×10^{-2}	1.9427
AAL1029	1.37×10^{-2}	9.64×10^{-3}	3.19×10^{-2}	2.37×10^{-2}	1.7655
AAL1340	1.43×10^{-2}	9.62×10^{-3}	2.43×10^{-2}	1.69×10^{-2}	1.3411
CPZ6068	1.64×10^{-2}	1.09×10^{-2}	8.99×10^{-2}	5.51×10^{-2}	3.8742
ASA770	1.66×10^{-2}	1.08×10^{-2}	7.49×10^{-2}	4.61×10^{-2}	3.2670
DAL1140	1.68×10^{-2}	1.11×10^{-2}	6.35×10^{-2}	3.88×10^{-2}	2.7846
Average	2.42×10^{-2}	1.42×10^{-2}	4.99×10^{-2}	3.15×10^{-2}	2.4747

information on the safety risk. Thirdly, the trained deep learning models offer new insights to view the trajectory prediction from different angles. By integrating the two models together, we not only improve the overall prediction accuracy, but also retain a longer-term prediction capability.

The following research directions need to be investigated further in the future. The impact of weather on the flight trajectory needs to be accounted for in an explicit manner. Currently such information is implicitly included through the historical data. However, for prediction regarding an ongoing flight, explicit use of available weather forecast information would be valuable. The challenging question is how to incorporate different types of weather and the probabilistic forecasting of the weather condition into the deep learning models. Another future direction is to develop a visualization system to show en-route flight safety at different levels, e.g., sector level, system level. Such a system will enable operators to identify the locations with higher safety risks in a timely manner, thereby improving the speed and effectiveness in maintaining the system safety. Thirdly, the hybrid model developed in this paper shows promising performance in predicting the trajectory of commercial flights within the USA airspace, it will be interesting to investigate how this model performs for flights in other airspaces around the world, especially international flights European flights as routes over Europe tend to be more erratic due to diverse regulations. Last but not the least, the bias varies over different flight IDs as shown in the last column of Table 5; it is worth enhancing the interpretability of the developed models to understand why the model has high prediction bias for some flight IDs than others.

Acknowledgments

The authors are grateful to the anonymous reviewers whose constructive comments have greatly improved this manuscript. The research reported in this paper was supported by funds from NASA University Leadership Initiative program (Grant No. NNX17AJ86A, Technical Monitor: Dr. Kai Goebel) through subcontract to Arizona State University (Principal Investigator: Dr. Yongming Liu). The support of FAA and Harris Corporation in accessing the SWIM data is appreciated. This work was conducted in part using the resources of the Advanced Computing Center for Research and Education (ACCRE) at Vanderbilt University, Nashville, TN. The support is gratefully acknowledged.

Appendix A. Posterior distribution using Monte Carlo dropout

Considering the approximate distribution $q_\theta(\omega)$, by minimizing the Kullback–Leibler (KL) divergence between the approximating distribution $q_\theta(\omega)$ and the true posterior distribution $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$ as formulated in Eq. (A.1), we identify the best parametric distribution to approximate the actual posterior distribution.

$$\mathrm{KL}(q_\theta(\omega) \| \, p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})) = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})} d\omega = \int q_\theta(\omega) \log \frac{q_\theta(\omega) p(\boldsymbol{X}, \boldsymbol{Y})}{p(\omega, \boldsymbol{X}, \boldsymbol{Y})} d\omega$$

$$= \log p(\boldsymbol{X}, \boldsymbol{Y}) - \underbrace{\int q_\theta(\omega) \log \frac{p(\omega, \boldsymbol{X}, \boldsymbol{Y})}{q_\theta(\omega)} d\omega}_{\text{Evidence Lower Bound}} \tag{A.1}$$

Since $p(\boldsymbol{X}, \boldsymbol{Y})$ is a constant, minimizing the KL divergence is equivalent to maximizing the log evidence lower bound (ELBO) with respect to θ [53]:

$$\mathrm{L}_{\mathrm{ELBO}}(\theta) = \int q_\theta(\omega) \log p(\boldsymbol{Y}|\, \boldsymbol{X}, \omega) d\omega - \mathrm{KL}(q_\theta(\omega) \| \, p(\omega)) \tag{A.2}$$

where the first term measures the likelihood of generating the observed data from the parameter value sampled from the variational distribution $q_\theta(\omega)$, and it encourages the densities with mass distributed around the region that explains the data better; the second term is the negative divergence between the variational density and the prior: the closer to the prior distribution, the better. The θ that maximizes the objective function formulated in Eq. (A.2) will result in a variational distribution q_θ that makes a good approximation for the posterior $p(\omega|\, \boldsymbol{X}, \boldsymbol{Y})$ while still being close

to the prior $p(\omega)$.

The integral in Eq. (A.2) is computationally intractable for all q, therefore we perform the Monte Carlo integration to approximate this quantity. Specifically, we sample $\hat{\omega}$ from $q_\theta(\omega)$, then perform one-step optimization with respect to θ to maximize the objective function formulated in Eq. (A.3). Next, we update $q_\theta(\omega)$, and draw a new sample $\hat{\omega}$ from the updated $q_\theta(\omega)$. By doing this again and again, we eventually converge to a θ that approximates the true posterior distribution $p(\omega|\boldsymbol{X}, \boldsymbol{Y})$ the best.

$$\hat{L}(\theta) = \log p(\boldsymbol{Y}|\boldsymbol{X}, \hat{\omega}) d\omega - KL(q_\theta(\omega) \| p(\omega)) \tag{A.3}$$

In neural networks, by defining q as a factorization over the weight matrices $\boldsymbol{W}_i$ across all the layers ($i = 1, \quad 2, \cdots, L$), we have [27, 51]:

$$q_\theta(\omega) = \prod_i q_{\boldsymbol{M}_i}(\boldsymbol{W}_i) \tag{A.4}$$

where the q of each $\boldsymbol{W}_i$ is defined as the product of the mean weight matrix $\boldsymbol{M}_i (\boldsymbol{M}_i = \text{mean}(\boldsymbol{W}_i))$ with the diagonal matrix of each q holding Bernoulli variables. Mathematically, we have [27, 51]:

$$\begin{aligned} &q_{\boldsymbol{M}_i}(\boldsymbol{W}_i) = \boldsymbol{M}_i \bullet \text{diag}([\boldsymbol{z}_{i,j}]) \\ &\boldsymbol{z}_{i,j} \sim \text{Bernoulli}(p_i) \\ &\boldsymbol{W}_i \sim q_{\boldsymbol{M}_i}(\boldsymbol{W}_i) \end{aligned} \tag{A.5c}$$

As shown in Eq. (A.5b), sampling the diagonal elements $\boldsymbol{z}$ from a Bernoulli distribution is identical to randomly setting the corresponding units of the neural network as zeros. See Ref. [40] for more details. Thus, Monte Carlo dropout can be used to approximate Bayesian inference in deep neural networks.

References

[1] International Air Transport Association (IATA), IATA forecasts passenger demand to double over 20 years, International Air Transport Association (IATA) http://www.iata.org/pressroom/pr/Pages/2016-10-18-02.aspx Accessed: 2019-02-12.

[2] X. Zhang, S. Mahadevan, Aircraft re-routing optimization and performance assessment under uncertainty, Decision Support Systems 96 (2017) 67–82.

[3] S. Sankararaman, I. Roychoudhury, X. Zhang, K. Goebel, Preliminary investigation of impact of technological impairment on trajectory-based operation, 17th AIAA Aviation Technology, Integration, and Operations Conference, 2017, p. 4488.

[4] X. Zhang, S. Mahadevan, Ensemble machine learning models for aviation incident risk prediction, Decision Support Systems 116 (2019) 48–63.

[5] Federal Aviation Administration, U.S. Department of Transportation, The Future of the NAS, https://www.faa.gov/nextgen/media/futureofthenas.pdf, (2016).

[6] Air Traffic Organization, Federal Aviation Administration, 2015 Safety Report, Accessed: 2019-08-05 https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/safety/media/2015_safety_report.pdf Accessed: 2019-08-05.

[7] J.L. Yepes, I. Hwang, M. Rotea, New algorithms for aircraft intent inference and trajectory prediction, Journal of Guidance, Control, and Dynamics 30 (2) (2007) 370–382.

[8] X. Prats, V. Puig, J. Quevedo, F. Nejjari, Multi-objective optimisation for aircraft departure trajectories minimising noise annoyance, Transportation Research Part C: Emerging Technologies 18 (6) (2010) 975–989.

[9] A de Leege, M. van Paassen, M. Mulder, A machine learning approach to trajectory prediction, AIAA Guidance, Navigation, and Control (GNC) Conference, 2013, p. 4782.

[10] C. Gong, D. McNally, A methodology for automated trajectory prediction analysis, AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004, p. 4788.

[11] M. Daigle, I. Roychoudhury, L. Spirkovska, K. Goebel, S. Sankararaman, J. Ossenfort, C.S. Kulkarni, Real-time prediction of safety margins in the national airspace, 17th AIAA Aviation Technology, Integration, and Operations Conference, 2017, p. 4388.

[12] A. Nuic, D. Poles, V. Mouillet, BADA: an advanced aircraft performance model for present and future ATM systems, International Journal of Adaptive Control and Signal Processing 24 (10) (2010) 850–866.

[13] X. Guan, R. Lv, L. Sun, Y. Liu, A study of 4D trajectory prediction based on machine deep learning, 2016 12th World Congress on Intelligent Control and Automation (WCICA), IEEE, 2016, pp. 24–27.

[14] S. Ayhan, H. Samet, Aircraft trajectory prediction made easy with predictive analytics, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 21–30.

[15] H. Wu, Z. Chen, W. Sun, B. Zheng, W. Wang, Modeling trajectories with recurrent neural networks, Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 3083–3090.

[16] R. Alligier, D. Gianazza, Learning aircraft operational factors to improve aircraft climb prediction: a large scale multi-airport study, Transportation Research Part C: Emerging Technologies 96 (2018) 72–95.

[17] C. Di Ciccio, H. Van der Aa, C. Cabanillas, J. Mendling, J. Prescher, Detecting flight trajectory anomalies and predicting diversions in freight transportation, Decision Support Systems 88 (2016) 1–17.

[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[19] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Networks 61 (2015) 85–117.

[20] M. Moradi Kordmahalleh, M. Gorji Sefidmazgi, A. Homaifar, A sparse recurrent neural network for trajectory prediction of Atlantic hurricanes, Proceedings of the Genetic and Evolutionary Computation Conference 2016, ACM, 2016, pp. 957–964.

[21] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social LSTM: human trajectory prediction in crowded spaces, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 961–971.

[22] Z. Ghahramani, Probabilistic machine learning and artificial intelligence, Nature 521 (7553) (2015) 452.

[23] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, (2015) arXiv preprint arXiv:1505.05424.

[24] Y. Han, Y. Deng, A hybrid intelligent model for assessment of critical success factors in high risk emergency system, Journal of Ambient Intelligence and Humanized Computing 9 (6) (2018) 1933–1953.

[25] X. Zhang, S. Mahadevan, X. Deng, Reliability analysis with linguistic data: an evidential network approach, Reliability Engineering & System Safety 162 (2017) 111–121.

[26] Y. Gal, Z. Ghahramani, Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference, (2015) arXiv preprint arXiv:1506.02158.

[27] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, International Conference on Machine Learning, 2016, pp. 1050–1059.

[28] L. Zhu, N. Laptev, Deep and confident prediction for time series at Uber, 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 2017, pp. 103–110.

[29] C. Leibig, V. Allken, M.S. Ayhan, P. Berens, S. Wahl, Leveraging uncertainty information from deep neural networks for disease detection, Scientific Reports 7 (1) (2017) 17816.

[30] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, et al., Apache Spark: a unified engine for big data processing, Communications of the ACM 59 (11) (2016) 56–65.

[31] M.G. Lozano, J. Schreiber, J. Brynielsson, Tracking geographical locations using a geo-aware topic model for analyzing social media data, Decision Support Systems 99 (2017) 18–29.

[32] A. Usmani, System wide information management, 2010 Integrated Communications Navigation and Surveillance Conference (ICNS), IEEE, 2010, pp. 1–17.

[33] V. Sze, Y.-H. Chen, T.-J. Yang, J.S. Emer, Efficient processing of deep neural networks: a tutorial and survey, Proceedings of the IEEE 105 (12) (2017) 2295–2329.

[34] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Cognitive Modeling 5 (3) (1988) 1.

[35] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.

[36] C. Olah, Understanding LSTM Networks, https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Accessed: 2019-02-12.

[37] R.M. Neal, Bayesian Learning for Neural Networks, 118 Springer Science & Business Media, 2012.

[38] D.M. Blei, A. Kucukelbir, J.D. McAuliffe, Variational inference: a review for statisticians, Journal of the American Statistical Association 112 (518) (2017) 859–877.

[39] I. Osband, J. Aslanides, A. Cassirer, Randomized prior functions for deep reinforcement learning, Advances in Neural Information Processing Systems, 2018, pp. 8617–8629.

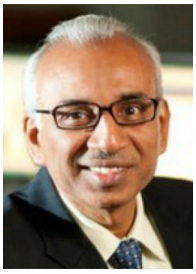
[40] Y. Gal, Uncertainty in Deep Learning, PhD thesis University of Cambridge, 2016.

[41] J.G. Shanahan, L. Dai, Large scale distributed data science using Apache Spark, Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 2323–2324.

[42] S. Salloum, R. Dautov, X. Chen, P.X. Peng, J.Z. Huang, Big data analytics on Apache Spark, International Journal of Data Science and Analytics 1 (3-4) (2016) 145–164.

[43] K. Hyukjin, Spark-XML, https://github.com/databricks/spark-xml Accessed: 2019-02-12.
[44] H. Yu, J. Wang, Z. Huang, Y. Yang, W. Xu, Video paragraph captioning using hierarchical recurrent neural networks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4584–4593.
[45] A. Graves, N. Jaitly, Towards end-to-end speech recognition with recurrent neural networks, International Conference on Machine Learning, 2014, pp. 1764–1772.
[46] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6645–6649.
[47] S. Liu, N. Yang, M. Li, M. Zhou, A recursive recurrent neural network for statistical machine translation, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 1 2014, pp. 1491–1500.
[48] J. Evermann, J.-R. Rehse, P. Fettke, Predicting process behaviour using deep learning, Decision Support Systems 100 (2017) 129–140.
[49] M. Liang, X. Hu, Recurrent convolutional neural network for object recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3367–3375.
[50] B. Singh, T.K. Marks, M. Jones, O. Tuzel, M. Shao, A multi-stream bi-directional recurrent neural network for fine-grained action detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1961–1970.
[51] Y. Gal, Z. Ghahramani, A theoretically grounded application of dropout in recurrent neural networks, Advances in Neural Information Processing Systems, 2016, pp. 1019–1027.
[52] K.R. Allendoerfer, S. Pai, F.J. Friedman-Berg, The complexity of signal detection in air traffic control alert situations, Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 52 SAGE Publications Sage, CA: Los Angeles, CA, 2008, pp. 54–58.
[53] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Xiaoge Zhang received his B.S. degree from Chongqing University of Posts and Telecommunications in 2011, and the M.S. degree from Southwest University in 2014, both in Chongqing, China; and Ph.D. from Vanderbilt University in May 2019, Nashville, TN. Currently, he works as a post-doctoral research scholar at Vanderbilt University, Nashville, TN, USA. From August to December in 2016, he interned at the National Aeronautics and Space Administration (NASA) Ames Research Center (ARC), Moffett Field, CA, working at the Prognostics Center of Excellence (PCoE) led by Dr. Kai Goebel. He was a recipient of the Chinese Government Award for Outstanding Self-financed Students Abroad in 2017. He has published more than 30 research papers in leading academic journals, such as IEEE Transactions on Cybernetics, Risk Analysis, IEEE Transactions on Reliability, Decision Support Systems, Information Sciences, Reliability Engineering and System Safety, Annals of Operations Research, and International Journal of Production Research, among others. His current research interests include uncertainty quantification, reliability assessment, machine learning, network optimization, and data analytics. He is a member of IEEE, INFORMS, and SIAM.



Sankaran Mahadevan is John R. Murray Sr. Professor of Engineering, and Professor of Civil and Environmental Engineering at Vanderbilt University, Nashville, Tennessee, where he has served since 1988. He is currently co-director of the Laboratory for Systems Integrity and Reliability (LASIR), and director of M. Eng. program in Risk, Reliability and Resilience Engineering. His research interests are in the areas of uncertainty quantification, model verification and validation, reliability and risk analysis, optimization under uncertainty, and system health monitoring, with applications to civil, mechanical and aerospace systems. His research has been extensively funded by NSF, NASA, FAA, DOE, DOD, DOT, NIST, GE, GM, Chrysler, Union Pacific, American Railroad Association, and the Sandia, Idaho, Los Alamos and Oak Ridge National Laboratories. His research contributions are documented in more than 600 publications, including two textbooks on reliability methods and 300 journal papers. He has directed 44 Ph.D. dissertations and 24 M.S. theses, and has taught several industry short courses on reliability and risk analysis methods. He is a Fellow of AIAA, EMI (ASCE), and PHM Society. His awards include the NASA Next Generation Design Tools award, SAE Distinguished Probabilistic Methods Educator Award, SEC Faculty Award, and best paper awards in the MORS Journal and the SDM and IMAC conferences. Professor Mahadevan obtained his B.S. from Indian Institute of Technology, Kanpur, M.S. from Rensselaer Polytechnic Institute, Troy, NY, and Ph.D. from Georgia Institute of Technology, Atlanta, GA.