Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)

Predictive Distribution of the Mass and Speed Profile to Improve Aircraft Climb Prediction

Richard Alligier
ENAC, Université de Toulouse, France

***Abstract*—Ground-based aircraft trajectory prediction is a major concern in air traffic control and management. A safe and efficient prediction is a prerequisite to the implementation of new automated tools.**

In current operations, trajectory prediction is computed using a physical model. It models the forces acting on the aircraft to predict the successive points of the future trajectory. Using such a model requires knowledge of the aircraft state (mass) and aircraft intent (thrust law, speed intent). Most of this information is not available to ground-based systems.

Focusing on the climb phase, we train neural networks to predict some of the unknown point-mass model parameters. These unknown parameters are the mass and the speed intent. For each unknown parameter, our model predicts a Gaussian distribution. This predicted distribution is a predictive distribution: it is the distribution of possible unknown parameter values conditional to the observed past trajectory of the considered aircraft.

Using this distribution, one can extract a predicted value and the uncertainty related to this specific prediction. Using a physical model like BADA, this distribution could be used to derive a probability distribution of possible future trajectory ([1]).

This study relies on ADS-B data coming from The OpenSky Network. It contains the climbing segments of the year 2017 detected by this sensor network. The 11 most frequent aircraft types are studied. The obtained data set contains millions of climbing segments from all over the world.

Using this data, we show that despite having an RMSE slightly larger than previously tested methods, the predicted uncertainty allows us to reduce the size of prediction intervals while keeping the same coverage probability. Furthermore, we show that the trajectories with a similar predicted uncertainty have an observed RMSE close to the predicted one.

The data set and the machine learning code are publicly available.

Keywords: aircraft trajectory prediction, BADA, mass, speed, machine learning, neural network

INTRODUCTION

Most applications in Air Traffic Control and Management (ATC/ATM) rely on a ground-based trajectory prediction. It will be even more true with new operational concepts [2, 3] envisioning trajectory-based operations. An accurate trajectory prediction is required for the new automated tools and algorithms implementing these concepts. Some of the most recent algorithms designed to solve ATM/ATC problems do require to test a large number of "what-if" alternative trajectories and it would be impractical to download them all from the aircraft. As an example, in [4] an iterative quasi-Newton method is used to find trajectories for departing aircraft, minimizing the noise annoyance. Another example is [5] where Monte Carlo simulations are used to estimate the risk of conflict between trajectories, in a stochastic environment. Some of the automated tools currently being developed for ATC/ATM can detect and solve conflicts between trajectories, using Genetic Algorithms ([6][1]), or Differential Evolution or Particle Swarm Optimization ([8]). In these conflict solving algorithms, each considered maneuver is associated to the trajectory predicted if such a maneuver was issued. If the trajectory prediction is bad, a large safety margin around the predicted trajectories will be taken. As a result, the only remaining conflict-free maneuvers might be the one associated to a large cost. With a good trajectory prediction, the safety margin around the predicted trajectories will be smaller. The set of conflict-free trajectories will be larger and might contain maneuvers of smaller cost.

Most trajectory predictors rely on a point-mass model to describe the aircraft dynamics. The aircraft is simply modeled as a point with a mass, and the second Newton's law is applied to relate the forces acting on the aircraft to the inertial acceleration of its center of mass. Such a model is formulated as a set of differential algebraic equations that must be integrated over a time interval in order to predict the successive aircraft positions, knowing the aircraft initial state (mass, current thrust setting, position, velocity, bank angle, etc.), atmospheric conditions (wind, temperature), and aircraft intent (thrust profile, speed profile, route). The Eurocontrol Base of Aircraft Data (BADA) project ([9]) implements such a physical model and provides default values for the models parameters.

In current operations, the trajectory is predicted by using the reference mass $\mathrm{mass}_{\mathrm{ref}}$ and the reference ($\mathrm{cas}_{1\mathrm{ref}}$, $\mathrm{cas}_{2\mathrm{ref}}$, $\mathrm{Mach}_{\mathrm{ref}}$) values from BADA. The latter values describe the speed profile of a climbing aircraft.

This paper focuses on the climbing phase because the unknown parameters have a great impact on the trajectory during this phase. In this paper, we apply machine learning methods to predict: the mass m and the speed profile parameters $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$. The predicted parameters will hopefully provide better trajectory predictions than the default BADA values. The predictive models are trained on historical data containing a large number of past flights collected over the first ten months of the year 2017. For each parameter, a Gaussian distribution is predicted $\mathcal{N}(\mu(x); \sigma(x))$ where x is a vector of features embedding all the information available about the considered climbing aircraft. Knowing x, each parameter is

[1]These algorithms are at the root of the strategic deconfliction through speed adjustments developed in the European ERASMUS project ([7]). A more recent application is the SESAR 4.7.2 (*Separation Task in En Route Trajectory-based Environment*) project, where lateral and vertical maneuvers are also used.

supposed to follow the predicted Gaussian distribution. One can interpret $\mu(x)$ as the predicted value and $\sigma(x)$ as the uncertainty on the predicted value.

The main contribution of this paper is to use machine learning on a large historical data set to predict distributions of the unknown parameters from the past points of a climbing aircraft. To our knowledge, in previous works, the methods used to predict distributions of the mass did not use historical data and those using historical data set were not able to scale to large data set.

The rest of the paper is organized as follows: Section I presents the context and the approach of this study. Section II describes the data used in this study. Section III explains how the sets of examples used in machine learning are built. Section IV details the machine learning method used, and the results are shown and discussed in Section V, before the conclusion.

I. Context

Some studies ([10, 11, 12]) detail the potential benefits that would be provided by additional or more accurate input data. In other works, the aircraft intent is formalized through the definition of an Aircraft Intent Description Language ([13, 14]) that could be used in air-ground data links to transmit some useful data to ground-based applications. All the necessary data required to predict aircraft trajectories might become available to ground systems someday. In the meantime different methods have been designed to obtain these input parameters from the data that is already available today.

Many recent studies ([15, 16, 17, 18, 19, 20]) used past trajectory points to estimate the aircraft mass using a total energy model such as BADA. All these methods adjust the mass to fit observed values of energy variation. In [21], a mass estimate is extracted from the down-linked Extended Projected Profile (EPP) with the aim to facilitate air-ground trajectory synchronization. [22] fits mean thrust setting profiles using mass estimation methods described in [18] and a set of flights. In all these studies, the methods provide only an estimate of the mass, they do not provide any information about the uncertainty related to this estimate.

[23, 24] propose a Bayesian approach to merge several mass estimates into a refined posterior probability distribution. It assumes that the estimates are independent and the error made on each estimate follows a given Gaussian. Then, assuming that the true mass follows a Gaussian prior, the posterior is also a Gaussian and can be obtained through simple calculation. In [25], the mass and the thrust setting are estimated altogether. A Gaussian noise is assumed on the position and velocity observed. An additive Gaussian noise is also assumed concerning the states evolution equations. Then, a numerical approximation of the posterior is computed using particle filter techniques. All these techniques do not take advantage of historical data as opposed to machine learning techniques.

Using Flight Data Recorder (FDR) historical data and machine learning, [26, 27] build a model that predicts the mass knowing the starting and ending speeds of the takeoff ground roll. Using Gaussian Process Regression (GPR), it predicts a Gaussian posterior distribution. However, this technique does not scale well with large historical data.

Using millions of ADS-B climbing segments, [28] builds models to predict the mass and the speed profile parameters ($\mathrm{cas}_{1_{\mathrm{ref}}}$, $\mathrm{cas}_{2_{\mathrm{ref}}}$, $\mathrm{Mach}_{\mathrm{ref}}$) from the past trajectory of a climbing aircraft. Using Gradient Boosting Machines (GBM), it does not provide any information about the uncertainty related to the computed prediction.

In this paper, using the same historical data than [28], we want to build a model that predicts a Gaussian distribution for each parameter. These distributions will be specific to the considered climbing aircraft. The method used must be able to process a large amount of data as opposed to GPR.

II. Data Used in this Study

The trajectory data used in this study are from The OpenSky Network ([29]). The OpenSky Network is a participatory sensor network of ADS-B sensors that covers mainly Europe and North-America. The data used in this study covers the year 2017. The augmented and sampled climbing segments used in this study are available at https://opensky-network.org/datasets/publication-data. This data set contains the 10 most frequent aircraft types according to [30]. These 10 aircraft types cover 63 % of the European air traffic according to [30]. Actually, in a recent ICAO 8643 document update, the E190 aircraft type designator has been split into two types namely E190 and E195. This leads us to consider 11 aircraft types. The description of this data set is more comprehensive in [28].

A. From Raw Trajectory Points to Sampled Climbing Segments

From The OpenSky Network, we have downloaded all the raw trajectory points of the year 2017 with a vertical rate superior or equal to 256 feet/min. These raw points are processed to obtain clean sampled climbing segments.

The points associated to the same aircraft are identified using the ICAO 24-bit values. As only positions with a positive vertical rate were downloaded, we only have points in climb phase. However, the sequence of points associated to one aircraft can contain several climbing segments. It may even contain different flights of the considered aircraft. We have to split this sequence of points into climbing segments. Moreover, we have decided that each climbing segment must contain at least one raw point every 30 seconds. The purpose of this requirement is to ensure the quality of the climbing segments we handle in this study.

Consequently, we have split the sequence of points into sub-sequences with no time hole superior to 30 seconds. These sub-sequences will be our climbing segments. Please note that two climbing segments can come from the same continuous climb if somehow no position update has been received within 30 seconds during this climb. Conversely, two different continuous climbs will give us two different climbing segments as the two continuous climbs are most likely like than 30 seconds apart.

These sub-sequences are then sampled using interpolation to obtain one point every 15 seconds.

B. Adding Relevant Information to our Data

Adding information to our data is a mandatory step as the climbing segments do not contain the aircraft type nor the weather for example. These two information are very important in trajectory prediction.

1) Aircraft Type, Aircraft Variant and Airline Operator: The aircraft type was identified using the ICAO 24-bit address in our segments. Using this address, the aircraft type was retrieved from several databases. For this purpose, an aircraft database was built using VirtualRadarServer[2] and its database writer plugin. If this database did not contain the ICAO 24-bit we were looking for, then we searched it in the World Aircraft Database [31].

In addition to the aircraft type, this database contains the aircraft variant and airline operator. When available, we have also added this information to our segments.

2) Weather: The Global Forecast System (GFS) was used to add the weather to our segments. More precisely, we have used the forecast files, not the analysis files, with a 1-degree grid. We have one weather grid every 3 hours.

3) Departure and Arrival airports: Using the callsign in our segments and the route database from FlightAirMap[3], the departure and arrival airports were identified.

C. Statistics on the Sampled Climbing Segments Used in Machine Learning

The number of climbing segments obtained for each aircraft type for the year 2017 are presented in the Table I. It only includes segments that last more than 750 seconds, as the others were discarded. All these segments will be used to train or test our models.

Table I: This table summarizes the number of climbing segments with a duration superior to 750 seconds. All these climbing segments will be used to train or test our models.

model	B738	A320	A319	A321	E195
count	1,344,709	1,340,691	564,308	596,749	68,965

model	E190	DH8D	B737	CRJ9	A332	B77W
count	39,534	27,867	149,065	27,370	109,534	123,622

Figure 1 plots the sampled climbing segments on a world map. In order to produce this figure, 331 millions aircraft positions were aggregated. With this figure, we can see that the five continents contains climbing segments. However, most of them are located in Europe and North-America. Africa contains the fewest climbing segments.

III. Building the Sets of Examples for Our Prediction Problem

Machine learning techniques use a set of (x, y) examples to build a model predicting y from x. This section describes how we obtain such a set of examples from the climbing segments. In our prediction problem, x is the information available at the time the prediction is computed and y is the mass and the speed profile $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$.

[2]http://www.virtualradarserver.co.uk/

[3]https://data.flightairmap.com/

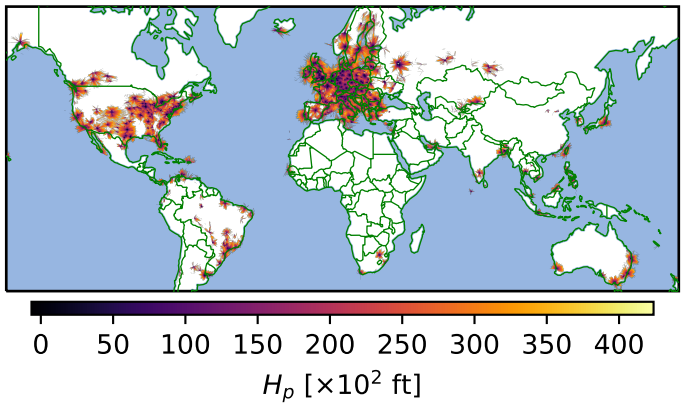

Figure 1: Climbing segments plotted on a world map. The mean altitude can be read from the color.

A. Extracting Trajectory Samples from One Climbing Segment

In this study, knowing the current position and $p = 9$ consecutive past points, we want to predict the future q points. In this context, a trajectory sample is defined by the current position, the p past points and the q future points. The trajectory samples will be used to train and evaluate our predictive models. These trajectory samples are built from the climbing segments. Actually, we build several trajectory samples from one climbing segment.

Considering one climbing segment with n points, a trajectory sample is built from $p+q+1$ consecutive points chosen among the n segment points. Hence, from one segment we build $n-p-q$ trajectory samples. Figure 2 illustrates two different trajectory samples (with $q = 40$) extracted from the same climbing segment.

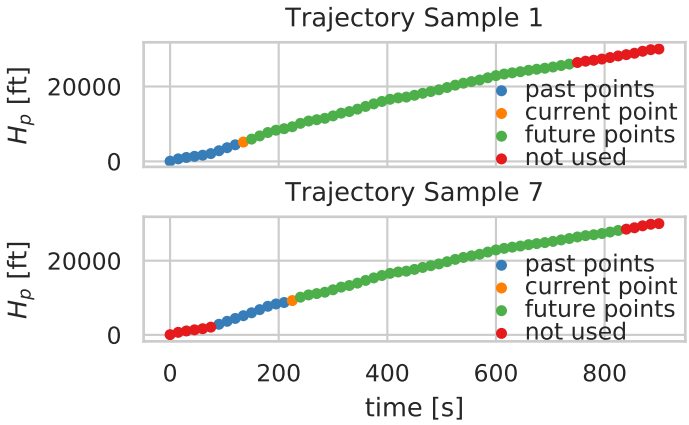

Figure 2: With $q = 40$, two different trajectory samples extracted from the same climbing segments.

B. Building One Example from One Trajectory Sample

This subsection describes how the mass and the speed profile $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$ can be extracted from one trajectory sample. These values will be the "y" of one example.

1) Adding the Mass: For each trajectory sample, the mass is estimated using the $q = 40$ future points. The method used to

extract the mass from these future points is the one described in [18]. This method assumes a max climb thrust. The mass is estimated by minimizing the difference between the modeled power and the observed energy variation.

We can compute the energy variation E_v on the past points using the derivative of the airspeed $\frac{dV_a}{dt}$ and the derivative of the pressure altitude $\frac{dH_p}{dt}$. Using a model of forces such as BADA, *Power* can be computed as a function of H_p, V_a, the temp T and the mass m. According to Newton's laws, these two quantities are equal. The mass to be learned will be the one minimizing the sum defined by equation (1) where $i = 10$ is the index of the current point in the trajectory sample and ff is the BADA function modeling the fuel consumption. As described in [18], minimizing such a sum can be done efficiently by finding the roots of a fourth degree polynomial.

$$m_{10} = \underset{m_{10}}{argmin} \sum_{i=10}^{10+q} \left(\frac{Power_i(H_{p_i}, V_{ai}, T_i, m_i)}{m_i} - \frac{E_{vi}}{m_i} \right)^2 \tag{1a}$$

$$\text{with } m_{i+1} = m_i - ff\,(V_a, H_p, T)\,(t_{i+1} - t_i) \tag{1b}$$

We have also applied this technique to estimate the mass on the past points. This mass estimated on past the points is added to the explanatory variable.

2) Adding the Speed Profile: The speed profile is modeled in BADA with three parameters cas_1, cas_2 and Mach. This speed profile specify the TAS V_a for a given altitude H_p and a temperature T. The aircraft climbs at a constant Calibrated AirSpeed (CAS) equal to cas_1 from 3,000 ft to 10,000 ft. Then, it accelerates till it reaches cas_2. It climbs at a constant CAS cas_2 till it reaches the Mach number Mach. Then it climbs at a constant Mach number.

We want to extract cas_1, cas_2 and Mach from the points in the trajectory sample. We can see that extracting a speed profile requires points from low altitude to high altitude. As a consequence, to extract the speed profile, we consider all the points in the climbing segment, not only the points in the trajectory sample. Hence, all the trajectory samples coming from the same climbing segment will have the same common $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$ minimizing the function e given by the equation (2).

$$e(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M}) = \sum_{i=1}^{n} \left(V_a(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M}; H_{p_i}, T_i) - V_{ai} \right)^2 \tag{2}$$

On Figure 3, a climbing segment and the fitted speed profile are plotted. This climbing segment was selected among the climbing segments with at least 3 points above the crossover altitude and 3 points below 10,000 ft. Among these segments, the climbing segment selected is the one with the median error. Thus, the speed profile accuracy in Figure 3 is quite representative of what can be obtained through this fitting process.

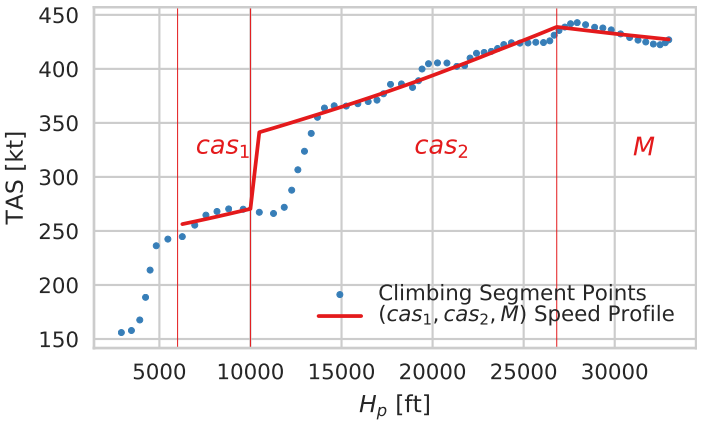

Figure 3: A climbing segment and the fitted (cas_1, cas_2, M) speed profile.

C. Explanatory Variables

This subsection describes the explanatory variables, the "x" variables, used to predict the the mass and the speed profile. We use 68 explanatory variables summarized in Table II.

These variables include information on the aircraft motion. They also include information on the weather: the temperature and wind at the current point, and the temperature every 1,000 m starting from the current altitude H_p to the altitude $H_p + 11{,}000$ m. This is useful as the temperature does not follow a temperature profile corresponding to an ISA atmosphere even when this temperature profile is corrected with a temperature differential ΔT. The temperature at $H_p = 0$ was also added. The temperature can influence the engine performance. It also impacts the geopotential altitude H between two given geopotential pressure altitude H_p (see BADA user manual). It also impacts the speed profile. As a consequence, depending on the temperature gradient, different energy share factor will be used, hence different climbing rate.

For each example, we also have categorical variables like the airline operator, the aircraft type variant, the departure and arrival airports and the day of the week. This last variable was included to make use of a possible seasonality. The month can also provide some insight on the seasonality however our data only covers one year so the month was not included.

When the departure and arrival airports were known we computed the trip distance between these two airports. This trip distance will provide information on the fuel load and hence the mass of the aircraft which affects the climb. The departure airport is used because the constraints that apply to the climbing aircraft might depend on the airport considered.

IV. Machine Learning

This section describes some useful machine learning notions and techniques. For a more detailed and comprehensive description of machine learning techniques, one can refer to [32, 33].

A. Statistical inference: learning from examples

Let us consider a set of n examples $S = (x_i, y_i)_{1 \leqslant i \leqslant n}$ coming from independent draws of the same joint distribution

Table II: A summary of the features used to predict the unknown parameters.

	feature description	count
categorical	departure and arrival airports	2
	aircraft type variant	1
	airline operator	1
	day of the week	1
	callsign	1
	ICAO 24 bit Mode-S address	1
numerical	distance between airports	1
	temperature at $H_p = 0$	1
	mass estimated on past points and error on past points	2
	track angle at the current point	1
	ground velocity at the current point	1
	north and east wind components	2
	longitude and latitude at the current point	2
	vertical speed at the current and past points	10
	altitude H_p at the current and past points	10
	airspeed V_a at the current and past points	10
	energy variation between the current and past points	9
	temperature from current altitude H_p to H_p+11,000 m	12

(X, Y). We want to deduce properties on the distribution $Y|X = x$ from the examples. This distribution is useful to obtain knowledge on y knowing x. Let us consider the probability density function of this distribution $p(y|X = x; \theta)$ where θ is an unknown parameter. Choosing a value for this parameter is choosing a model for $Y|X = x$.

We choose θ maximizing $p(S \mid \theta)$, the probability to generate the samples S for a given θ. It is the *maximum likelihood* estimate. The expression of $p(S|\theta)$ is easy to obtain:

$$p(S|\theta) = \prod_{i=1}^{n} p(y_i|X = x_i; \theta)p(x_i)$$

As the $p(x_i)$ are constants, maximizing $p(S|\theta)$ is minimizing the negative log-likelihood NLL:

$$\mathrm{NLL}(\theta; S) = -\sum_{i=1}^{n} \log(p(y_i|X = x_i; \theta))$$

Assuming $Y|X = x \sim \mathcal{N}(\mu(x; \theta), \sigma(x; \theta))$, this expression becomes:

$$\mathrm{NLL}(\theta; S) = \sum_{i=1}^{n} \frac{1}{2} \left(\frac{y_i - \mu(x_i; \theta)}{\sigma(x_i; \theta)} \right)^2 + \log \sigma(x_i; \theta) + \log \sqrt{2\pi}$$

The value $\mu(x; \theta)$ is the predicted mean for Y and the value $\sigma(x; \theta)$ is the predicted variance. It provides a precious information on the uncertainty of the prediction.

As a side note, if we assume that the variance is a constant value $\sigma(x; \theta) = \sigma_0$ (homoscedasticity), minimizing the NLL is equivalent to minimizing the mean squared error.

B. Predictive Uncertainty

Gaussian Process Regression ([34]) is a powerful non-parametric framework that handles some sort of prior probability over functions. Using this prior and the Bayesian formalism, this framework naturally derive a Gaussian distribution for $Y|X = x$. The exact computation of such a model requires $\mathcal{O}(n^3)$ operations which might be intractable for large data set like the one we use.

With the recent successes of the neural networks in several domains, some works ([35, 36]) introduce simple modifications to obtain both the predicted value and the predicted uncertainty. Ideally, the predicted value shall be equal to $\mathbb{E}[Y|X = x]$ whereas the predicted uncertainty shall be equal to $\mathbb{V}\mathrm{ar}[Y|X = x]$.

In [36], the neural network has two output vectors, the vector $\mu(x; \theta)$ that shall predict $\mathbb{E}[Y|X = x]$ and the vector $\sigma^2(x; \theta)$ that shall predict $\mathbb{V}\mathrm{ar}[Y|X = x]$. This network is trained by minimizing the negative log-likelihood NLL. Actually, the final model is not a single neural network but an ensemble of m networks. These networks are obtained by using a different random initialization with the same architecture and training set. As a consequence, with these m networks, for a given x, we obtain m predicted values and uncertainties: $\mu_i(x)$ and $\sigma_i(x)$ with $i \in [\![1; m]\!]$. In order to combine these predictions into one, we consider that $Y|X = x$ follows a mixture of m Gaussian distributions $\mathcal{N}(\mu_i(x), \sigma_i(x))$ with similar mixture weights. Then the predicted value and uncertainty are the mean and variance of this mixture and they can be computed using a simple formula combining all the predicted $\mu_i(x)$ and $\sigma_i(x)$.

C. Method Used in this Study

In this study we used the method developed in [36]. We use a fully connected feed forward network with several hidden layers. The Figure 4 depicts the architecture of this neural network (NN) where the green blocks are vectors and the red blocks are functions applied on these vectors. The activation function is a LeakyReLU function ([37]). The architecture of this NN is pretty standard except we have added a softplus[4] function on some components of the output vector and we have used embeddings to encode categorical inputs.

The softplus function is used to always obtain positive values for the predicted standard deviations.

Each categorical variable such as the callsign must be encoded into a vector of floating point numbers. This is done by using *embeddings*. One *embedding* maps each categorical value of a categorical variable to one vector of weights. If we encode n categorical values with vectors of d components then we have nd weights for the *embedding* of the considered categorical variable. These weights are randomly initialized and then trained by the optimization procedure just like the other weights of the network. The size of the vectors d is an hyper-parameter of the NN. This approach has been successfully used in [38] to predict the destination of a taxi based on the beginning of its trajectory.

To improve the performance, the training process uses *dropout* ([39]) and *batch normalization* ([40]) blocks. The *batch normalization* blocks are inserted after the LeakyReLU blocks. The *dropout* blocks are inserted after the *embedding* blocks.

[4]The softplus function is $x \mapsto \log(1 + \exp x)$. This function always returns a strictly positive value.

The *dropout* blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly chosen sub-network is used to compute the prediction and hence receive the weights updates. Conceptually, an ensemble of networks is trained altogether in an efficient way. After the training phase, the whole network is used to compute the prediction on new data.

The *batch normalization* blocks are used as a reparametrization method improving the optimization process. The gradient descent technique relies on a first-order approximation of the loss as a function of the weights. As a first-order approximation, it hides the interaction between the weights. The *batch normalization* block aims to reduce the interaction between the weights of the different layers. To do so, all the weights of the layers before one *batch normalization* block will have no impact on the mean and variance of the vector returned by the *batch normalization* block: the mean and variance of the returned vector are only controlled by two additional weights inside the *batch normalization* block.

The training phase consists in finding the weights minimizing the negative log-likelihood NLL. It is done using AdamW, a gradient descent method with adaptive learning rate and weight decay ([41]).

The initial learning rate is found via the search method described in [42]. The remaining hyper-parameters are the learning rate decay, the weight decay, the number of hidden layers, the number of hidden units for each hidden layers and the dropout rate. We tested 200 different sets of hyper-parameters. The tested hyper-parameters are randomly drawn. For instance, the number of hidden layers is drawn inside $[\![1; 10]\!]$ using a discrete uniform distribution. Such a random search is empirically and theoretically more efficient than a grid search ([43]). For each hyper-parameter, the neural network is trained on trajectories from January to August. Then it is tested on the trajectories from September to October, and the hyper-parameters having the best result on these trajectories will be the chosen one. Then using this selected hyper-parameters, the final model is the one trained on the trajectories from January to October.

All the code is implemented using the PyTorch library.

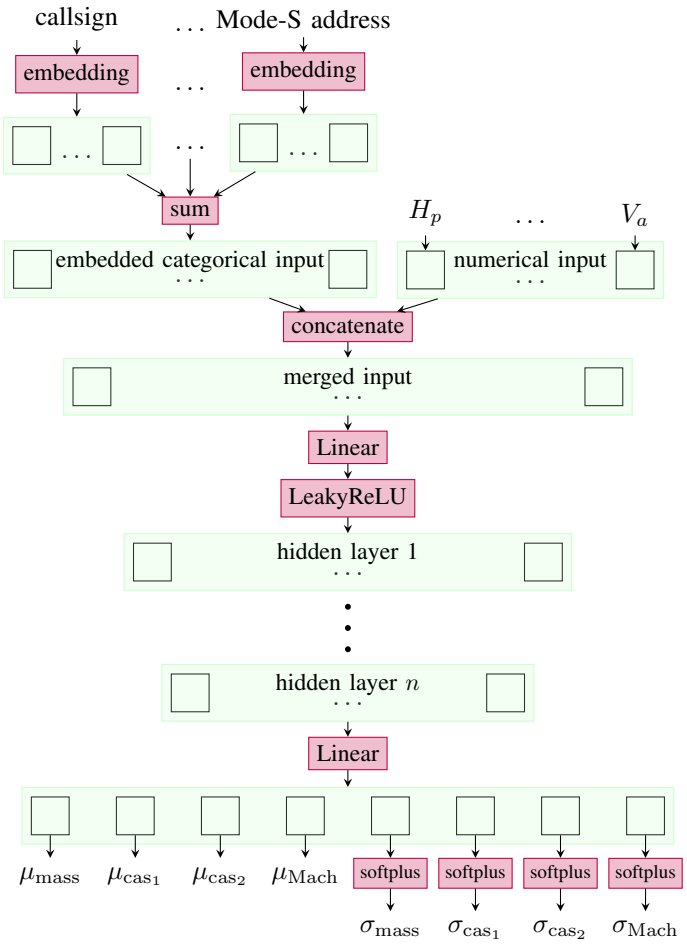

Figure 4: Architecture of the neural network we used. The manipulated vectors are in green whereas the operations applied to these vectors are in red. The input is at the top, the output is at the bottom.

V. Results

All the results presented in this section have been computed on data not used in the model building process. These results have been computed from all the trajectories in the months November and December of the year 2017. The ten first months of 2017 were used to build the predictive models. The *training set* uses trajectories recorded from January to August, the *validation set* use trajectories recorded in September and October and finally the *test set* use trajectories recorded in November and December. It is a simple hold-out validation.

The k-fold cross-validation usually provides a better assessment of the generalization error than a simple hold-out validation, nevertheless we chose the second approach here, for very specific reasons. The distribution of the trajectories might change through time if for instance new procedures are applied at a specific airport. Using a cross-validation that randomly places the examples in the folds will produce folds with the same distribution. It will mask the non-stationarity of the problem we are studying and the performance evaluation obtained will be too optimistic. For this reason, we chose a more practical approach, and decided that the model should be trained on a given period of time, and then tested on a later period of time, as would actually happen if the method was used in operations. If this performance evaluation is biased, it will be pessimistically biased.

All the statistics in this section have been computed on the *test set*, the trajectories recorded in November and December.

A. Prediction of the Mass and Speed Profile

The predicted parameters are the predicted values $(\mu_{\text{mass}}(x), \mu_{\text{cas}_1}(x), \mu_{\text{cas}_2}(x), \mu_{\text{Mach}}(x))$. These predicted values are compared with the "true" parameters extracted from the future trajectory. We will compare the neural network (NN) approach to the Gradient Boosting Machines (GBM) approach method. This latter has been tested in [28]. Compared with mean values, the GBM method have typically reduced the RMSE by 56 %, 49 %, 39 % and 15 %

respectively for the mass, cas_1, cas_2 and Mach.

The Table III gives the RMSE obtained with the method NN presented in this paper and GBM. The RMSE relative difference between NN and GBM do not exceed 12 %. Overall, NN performs slightly worse than GBM with a larger RMSE. This was somewhat expected as deep neural networks typically underperform GBM on many tabular-dataset learning tasks ([44]).

Table III: The statistics computed are the RMSE where the error is the difference between the predicted parameter and the parameter extracted from the future trajectory. It is computed on the *test set* consisting in trajectory samples from months November and December.

factor	mass [kg]		cas_1 [kt]		cas_2 [kt]		Mach [-]	
method	GBM	NN	GBM	NN	GBM	NN	GBM	NN
A319	2362	2387	8.31	8.76	9.50	9.58	0.0174	0.0175
A320	1929	1953	10.10	10.58	8.85	8.92	0.0197	0.0199
A321	2212	2234	8.42	8.64	9.48	9.49	0.018	0.0181
A332	8014	8241	9.66	9.92	7.35	7.36	0.0229	0.0228
B737	2511	2558	6.47	6.97	8.20	8.21	0.013	0.013
B738	2508	2532	8.44	9.17	7.84	7.89	0.0148	0.0148
B77W	10742	10621	7.11	7.50	5.47	5.51	0.0153	0.0154
CRJ9	1294	1283	9.10	9.24	7.44	7.42	0.0202	0.0205
DH8D	738	720	6.95	7.04	11.53	11.43	0.0236	0.0233
E190	2539	2604	8.22	8.61	7.09	7.00	0.0199	0.0199
E195	2126	2134	7.59	7.65	7.45	7.37	0.025	0.025

B. Prediction Interval

Alongside the predicted factor $\mu(x)$, one might want to have a value quantifying the uncertainty concerning the predicted value. In our method, this uncertainty is quantified by the predicted standard deviation $\sigma(x)$. The relevance of this predicted uncertainty is difficult to evaluate. The "ground truth" uncertainty is not available in our data-set as opposed to the "ground truth" value to be predicted y.

The predicted uncertainty $\sigma(x)$ can be used to build a prediction interval $I_\gamma(x)$ that should contain the true parameter with a probability γ:

$$\mathbb{P}\left(Y \in I_\gamma\left(x\right) | X = x\right) = \gamma. \tag{3}$$

Assuming that $Y|X = x \sim \mathcal{N}\left(\mu(x), \sigma(x)\right)$, the prediction interval $I_\gamma(x)$ can be built as an interval centered on the mean $\mu(x)$ with a size proportional to the standard deviation $\sigma(x)$. Hence the interval can be defined as:

$$I_\gamma(x) = [\mu(x) - r_\gamma \sigma(x); \mu(x) + r_\gamma \sigma(x)]$$

with r_γ a value that depends only on the probability γ chosen. For instance, in order to match the probability $\gamma = 0.95$, $r_\gamma = 1.96$ is chosen.

In order to test that the predicted intervals satisfy the equation (3), we can compute the Prediction Interval Coverage Probability (PICP): $\mathrm{PICP}_\gamma = \frac{1}{n} \sum_{(x,y)\in \text{test set}} \mathbb{1}_{I_\gamma(x)}(y)$, where $\mathbb{1}_{I_\gamma(x)}(y) = 1$ if $y \in I_\gamma(x)$ and 0 otherwise.

The obtained PICP are presented in Table IV. For $\gamma = 0.90$, the PICP is superior to γ for the vast majority of the aircraft and parameters considered. For $\gamma = 0.95$, the situation is more complex. For the mass and cas_1, the PICP is superior to γ for most aircraft whereas for cas_2 and Mach, the PICP is inferior to γ for most aircraft.

For safety purposes, having the PICP superior to γ is more desirable than the other way around. However, having the PICP significantly smaller or larger than γ is a problem as it is not expected from the theory. In our case, this might be explained by the fact that the prediction intervals are built using the assumption that the standardized error $z = \frac{y-\mu(x)}{\sigma(x)}$ follows a normal law $\mathcal{N}(0, 1)$. However, this assumption is not true: the z distribution has a slightly thicker tail than the normal distribution. This assertion is supported by the standardized kurtosis of z ranging from 0.2 to 42.6 depending on the considered aircraft and parameter.

Table IV: The statistics computed are the PICP_γ for $\gamma = 0.90$ and $\gamma = 0.95$.

factor	mass [%]		cas_1 [%]		cas_2 [%]		Mach [%]	
γ	0.90	0.95	0.90	0.95	0.90	0.95	0.90	0.95
A319	92.0	95.5	93.3	95.7	91.0	93.6	90.6	94.1
A320	92.1	95.7	93.4	95.8	91.1	93.8	91.5	94.4
A321	91.5	95.5	93.3	95.6	90.5	93.8	91.5	94.4
A332	92.8	95.9	93.5	95.5	91.7	94.6	91.8	94.3
B737	91.0	94.7	93.8	96.1	91.5	94.5	90.9	94.1
B738	91.8	95.3	93.7	96.1	91.7	94.7	92.2	94.9
B77W	91.9	95.5	94.5	96.8	91.6	94.0	90.9	93.9
CRJ9	90.3	94.5	92.1	94.8	90.8	93.8	92.3	95.5
DH8D	88.8	93.6	92.7	95.7	88.7	93.1	88.4	93.8
E190	88.1	92.4	92.3	94.8	91.0	94.1	91.4	94.8
E195	90.9	94.9	92.1	94.5	90.4	93.7	90.2	94.5

Roughly speaking, the PICP is competing with the interval size, the larger the interval is, the larger the PICP is. We want PICP large and the interval small. With our method NN, the size of the interval depends on the considered x. Let us compare this method to one for which no information are extracted from x: the interval size will be the same for all examples (x, y). Let us consider the GBM method, for each aircraft and parameters, we compute s such that:

$$P(Y \in [y_{GBM} - s; y_{GBM} + s] \,|X = x) = \mathrm{PICP}_\gamma, \tag{4}$$

where PICP_γ is the PICP obtained with NN and y_{GBM} is the predicted value by GBM. The interval $[y_{GBM} - s; y_{GBM} + s]$ will be the one predicted by GBM.

With these choices, we can compare the size of intervals that have the same PICP for NN and GBM. The intervals computed with NN will vary in size depending on x whereas the ones computed with GBM will have the same size.

Table V presents the mean size of the intervals. We have seen in Section V-A that NN have a slightly larger RMSE than GBM. Interestingly enough, the mean size of the intervals predicted by NN is smaller than the one predicted by GBM. Compared with GBM, the interval mean size is reduced by 4 % on average for the mass. This reduction is larger for the speed profile parameters with an average reduction of 31 %, 16 % and 6 % for cas_1, cas_2 and Mach respectively.

As opposed to the mass, the airspeed is a quantity directly transmitted through ADS-B. Thus, if the aircraft is in the cas_1 phase then the airspeed in the input x corresponds to the airspeed value cas_1. When plotted against the altitude, the $\sigma_{\mathrm{cas}_1}(x)$ is usually very low when the altitude is inside the

cas_1 phase. The same goes for the cas_2 and Mach variables. For this reason, the mean interval reduction is greater for the speed profile parameters than for the mass. The cas_1 phase is only delimited by the altitude whereas the delimitation of the cas_2 and Mach phases is less clear. This might explain why the reduction for cas_1 is larger than the one for cas_2 and Mach.

Table V: Mean size of the predicted interval for $\gamma = 0.90$. The intervals built with NN and GBM have same PICP.

factor	mass [kg]		cas_1 [kt]		cas_2 [kt]		Mach [-]	
method	GBM	NN	GBM	NN	GBM	NN	GBM	NN
A319	7435	7176	31.02	21.93	32.13	26.64	0.0554	0.051
A320	6435	6290	40.29	26.61	29.62	25.16	0.0644	0.0588
A321	7424	7258	29.29	20.38	32.79	26.78	0.0582	0.0543
A332	28260	26712	38.65	23.98	25.41	21.33	0.0743	0.0663
B737	7594	7617	23.47	17.11	29.07	23.63	0.0405	0.0385
B738	8202	7837	34.02	21.99	28.03	22.72	0.0468	0.0446
B77W	34892	32667	28.99	16.96	17.66	15.96	0.0452	0.045
CRJ9	4142	4039	32.17	22.13	24.49	20.46	0.072	0.0665
DH8D	2217	2129	26.06	18.11	37.30	32.99	0.074	0.0714
E190	7499	6834	29.33	22.08	24.80	20.01	0.0643	0.0621
E195	7127	6703	26.52	19.42	25.00	20.62	0.0792	0.0764

C. The Observed RMSE is Close to the Predicted $\sigma(x)$

In the previous subsection, we have computed statistics averaged over the whole test set. In this subsection, we want to compute statistics conditionally on the predicted $\sigma(x)$. Specifically, we want to empirically verify that for each $\sigma > 0$, $\mathbb{E}[(Y - \mu(X))^2 \mid \sigma(X) = \sigma] = \sigma^2$.

Let us consider $S(\sigma)$ a subset of the test set containing examples with $\sigma(x)$ similar to a given σ for the considered aircraft and parameter:

$$S(\sigma) = \{(x, y) \mid (x, y) \in \text{test set}, |\sigma(x) - \sigma| < \varepsilon_\sigma\}, \quad (5)$$

where ε_σ is used to control the size of $S(\sigma)$. For each σ, the ε_σ is chosen in order to have 1 % of the test set inside $S(\sigma)$.

The RMSE computed on the error made for the examples in $S(\sigma)$ should be close to σ. The Figures 5 and 6 are helpful to investigate this matter. Each figure contains two plots, the bottom plot is the distribution of $\sigma(x)$ for x in the test set and the top plot is the observed RMSE of the examples in $S(\sigma)$ as a function of σ. This curve is, in theory, close to the "y=x" red curve. For both figures, it is the case except for very high[5] σ. Nevertheless, for a very high σ, the associated RMSE is also very high.

Roughly speaking, in order to be informative, the distribution of the $\sigma(x)$ must be as spread as possible. If $\sigma(x)$ is the same for all the x then it will not provide any information.

Concerning the parameter cas_1, with the bottom plot of the Figure 6, we can see that there is a peak of density for low σ_{cas_1} value. This means that, for a large number of examples, our neural network is able to identify situation where the expected RMSE is very low, much lower than the RMSE on the whole test set. This observation is valid for cas_2 and for Mach to a lesser extent.

[5]For very high σ, ε_σ is very large in order to have 1 % of the data inside $S(\sigma)$ and the mean of $\{\sigma(x) \mid (x, y) \in S(\sigma)\}$ is close to $\sigma - \varepsilon_\sigma$ and far from σ.

Concerning the mass, with the bottom plot of the Figure 5, we can see that there is a large peak around the RMSE of the whole test set. Our neural network is less able to identify situations where the expected RMSE is low or high. Again, as said before, the past airspeed and altitude are included in x, making the prediction of cas_1 quite certain if the altitude is in the range of the cas_1 phase. For the mass, it is more difficult to identify situation where the predicted value will be certain or uncertain.

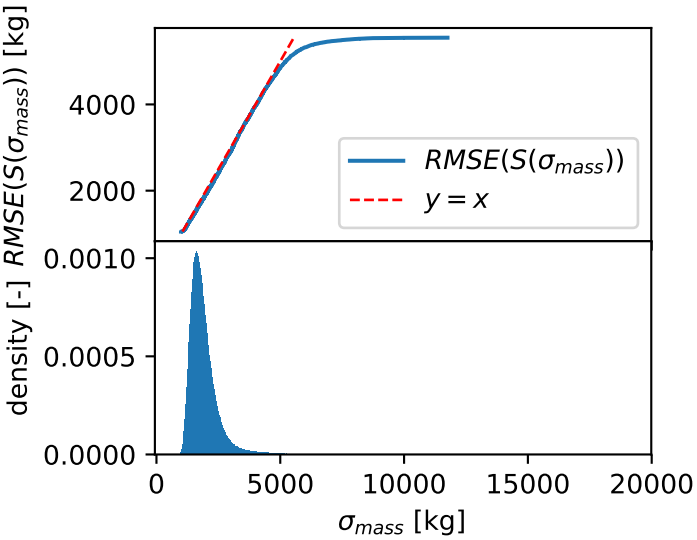

Figure 5: For the A320, the bottom plot is the distribution of $\sigma_{\text{mass}}(x)$ and the top plot is the RMSE of the examples in $S(\sigma_{\text{mass}})$ as a function of σ_{mass}.

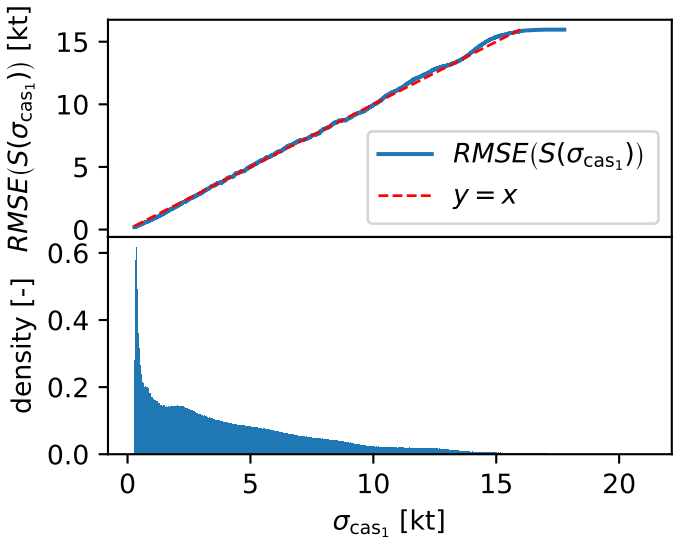

Figure 6: For the A320, the bottom plot is the distribution of $\sigma_{\text{cas}_1}(x)$ and the top plot is the RMSE of the examples in $S(\sigma_{\text{cas}_1})$ as a function of σ_{cas_1}.

D. Qualitative Analysis of Some Trajectories

In this subsection we investigate situations where a low or high uncertainty has been predicted and we also investigate situations where a low uncertainty is wrongfully predicted.

For all the aircraft types, the situations associated with the lowest predicted uncertainty for the mass ($\sigma_{\mathrm{mass}}(x)$) are the ones with an energy variation decreasing smoothly with the altitude. This is the typical energy variation profile that can be obtained by using BADA. The situations associated with the highest predicted uncertainty for the mass are the ones where the energy variation follows an irregular profile and/or the energy variation is much lower or much higher than the typical profile.

We now investigate the situations for which the error made is high compared with the predicted $\sigma(x)$. These situations are the ones where the observed y had a low probability assuming that this y was drawn from $\mathcal{N}(\mu(x), \sigma(x))$. Among these trajectories, for the mass parameter, the trajectories with a constant low ROCD climb phase are over-represented. For these trajectories, the NN wrongfuly predict a low uncertainty at the separation between the two phases: all the past points in x_t[6] are inside the "typical ROCD" phase whereas all the future points are inside the constant low ROCD phase. As the time t increases, the constant low ROCD points are included in x_t, resulting in an appropriate increase of the predicted uncertainty.

The NN does not anticipate the shift from a typical climb to a constant low ROCD climb and predicts a typical mass with a good confidence. However, these constant low ROCD phases are associated with a low energy rate and hence a heavy mass as we have assumed a max climb thrust. It is not surprising that the NN is unable to anticipate this type of climb shift. It is likely that there are no obvious clues inside past points that a type of climb shift will occur.

Conclusion

In this study we have tested machine learning methods using millions of climbing segments coming from The OpenSky Network. These climbing segments were completed with weather forecasts, aircraft types and variants, departure and arrival airports, estimated masses and speed profiles. The filtered and augmented data set is available at https://opensky-network.org/datasets/publication-data. The machine learning code is available at https://github.com/richardalligier/atm2019. Inside the ATM trajectory prediction community, we hope that sharing the data set and the machine learning code will enable scientifically sound comparisons based on the exact same data set.

Using this data set, we used an ensemble of neural networks to predict distributions for the parameters of a climbing aircraft: the mass and the $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$ speed profile values. These predictive distributions are Gaussian distributions $\mathcal{N}(\mu(x); \sigma(x))$ where x is all the information we have about the considered aircraft at the time the prediction is computed.

The RMSE associated with the predicted values $\mu(x)$ are slightly larger than the one observed using GBM ([28]). Interestingly enough, despite a slightly larger RMSE, the mean size of the prediction interval provided by the neural networks is slightly smaller than the one built with GBM, for the same actual coverage probability. For cas_1 and cas_2, the mean size is reduced by 31 % and 16 % respectively.

It has been demonstrated that the examples with a similar predicted $\sigma(x)$ are associated with an observed RMSE close to $\sigma(x)$. This can be useful to decide whether a prediction can be trusted or not.

These distributions could also be used to feed a method that convert distributions on aircraft parameters to distributions on future aircraft trajectory. Such a method is described in [1].

In future works, it could be interesting to build predictive distributions that do not assume a Gaussian distribution nor that the unknown parameters are independent from each other.

[6] At the time t the prediction is computed, the vector x contains the 10 past points . To make explicit the fact that these past points changes as t changes, we index x by t.

Acknowledgment

We thank The OpenSky Network team who granted us access to their wonderful database. We are also grateful that they host the data set that has been filtered and augmented with our algorithms so others can work easily on the same data set. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Authors

Richard Alligier received his Ph.D. (2014) degree in Computer Science from the "Institut National Polytechnique de Toulouse" (INPT), his engineer's degree (IEEAC, 2010) from the french university of civil aviation (ENAC) and his M.Sc. (2010) in computer science from the University of Toulouse. He is currently assistant professor at the ENAC in Toulouse, France.

References

[1] Enrique Casado, Marco La Civita, Miguel Vilaplana, and Euan W. McGookin. Quantification of aircraft trajectory prediction uncertainty using polynomial chaos expansions. In *IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, 2017.

[2] SESAR Consortium. Milestone Deliverable D3: The ATM Target Concept. Technical report, 2007.

[3] Harry Swenson, Richard Barhydt, and Michael Landis. Next Generation Air Transportation System (NGATS) Air Traffic Management (ATM)-Airspace Project. Technical report, National Aeronautics and Space Administration, 2006.

[4] Xavier Prats, Vincenç Puig, Joseba Quevedo, and Fatiha Nejjari. Multi-objective optimisation for aircraft departure trajectories minimising noise annoyance. *Transportation Research Part C*, 18(6):975–989, 2010.

[5] Georgios Chaloulos, Eva Crück, and John Lygeros. A simulation based study of subliminal control for air traffic management. *Transportation Research Part C: Emerging Technologies*, 18(6):963 – 974, 2010. Special issue on Transportation Simulation Advances in Air Transportation Research.

[6] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[7] Fabrice Drogoul, Philippe Averty, and Rosa Weber. Erasmus strategic deconfliction to benefit sesar. In *Proceedings of the 8th USA/Europe Air Traffic Management R&D Seminar*, June-July 2009.

[8] Charlie Vanaret, David Gianazza, Nicolas Durand, and Jean-Baptiste Gotteland. Benchmarking conflict resolution algorithms. In *International Conference on Research in Air Transportation (ICRAT), Berkeley, California, 22/05/12-25/05/12*, page (on line), http://www.icrat.org, may 2012. ICRAT.

[9] Vincent Mouillet. User manual for base of aircraft data (bada) rev.3.14. Technical report, EUROCONTROL, 2017.

[10] Peter Martin and Georges Mykoniatis. Study of the acquisition of data from aircraft operators to aid trajectory prediction calculation. Technical report, EUROCONTROL Experimental Center, 1998.

[11] ADAPT2. aircraft data aiming at predicting the trajectory. data analysis report. Technical report, EUROCONTROL Experimental Center, 2009.

[12] Richard A. Coppenbarger. Climb trajectory prediction enhancement using airline flight-planning information. In *AIAA Guidance, Navigation, and Control Conference*, 1999.

[13] Javier López-Leonés, Miguel A. Vilaplana, Edouardo Gallo, Francisco A. Navarro, and Carlos Querejeta. The aircraft intent description language: A key enabler for air-ground synchronization in trajectory-based operations. In *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference*. DASC, 2007.

[14] Javier Lopes-Leonés. *The Aircraft Intent Description Language*. PhD thesis, University of Glasgow, 2007.

[15] Charles A Schultz, David Thipphavong, and Heinz Erzberger. Adaptive trajectory prediction algorithm for climbing flights. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, August 2012.

[16] David P Thipphavong, Charles A Schultz, Alan G Lee, and Steven H Chan. Adaptive algorithm to improve trajectory prediction accuracy of climbing aircraft. *Journal of Guidance, Control, and Dynamics*, 36(1):15–24, 2012.

[17] Young S. Park and David P. Thipphavong. Performance of an Adaptive Trajectory Prediction Algorithm for Climbing Aircraft. In *2013 Aviation Technology, Integration, and Operations Conference*, page (on line). 08, Aug 2013.

[18] Richard Alligier, David Gianazza, Mohammad Ghasemi Hamed, and Nicolas Durand. Comparison of Two Ground-based Mass Estimation Methods on Real Data (regular paper). In *International Conference on Research in Air Transportation (ICRAT), Istambul, 26/05/2014-30/05/2014*, page (on line), http://www.icrat.org, may 2014. ICRAT.

[19] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. Modeling and inferring aircraft takeoff mass from runway ads-b data. In *7th International Conference on Research in Air Transportation*, 2016.

[20] Mevlüt Uzun and Emre Koyuncu. Data-driven trajectory uncertainty quantification for climbing aircraft to improve ground-based trajectory prediction. *Anadolu Üniversitesi Bilim Ve Teknoloji Dergisi A-Uygulamalı Bilimler ve Mühendislik*, 18(2):323–345, 2017.

[21] Jesper Bronsvoort, Greg McDonald, Mike Paglione, Christina M Young, Jean Boucquey, Joachim K Hochwarth, and Eduardo Gallo. Real-time trajectory predictor calibration through extended projected profile down-link. In *Eleventh USA/Europe Air Traffic Management Research and Development Seminar*, 2015.

[22] Richard Alligier, David Gianazza, and Nicolas Durand. Learning the aircraft mass and thrust to improve the ground-based trajectory prediction of climbing flights. *Transportation Research Part C: Emerging Technologies*, 36:45 – 60, 2013.

[23] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. Bayesian inference of aircraft initial mass. In *Proceedings of the 12th USA/Europe Air Traffic Management Research and Development Seminar*. FAA/EUROCONTROL, 2017.

[24] Junzi Sun, Joost Ellerbroek, and Jacco M. Hoekstra. Aircraft initial mass estimation using bayesian inference method. *Transportation Research Part C: Emerging Technologies*, 90:59 – 73, 2018.

[25] Junzi Sun, Henk AP Blom, Joost Ellerbroek, and Jacco M Hoekstra. Aircraft mass and thrust estimation using recursive bayesian method. page (on line), June 2018.

[26] Yashovardhan S Chati and Hamsa Balakrishnan. Statistical modeling of aircraft takeoff weight. 2017.

[27] Yashovardhan S Chati and Hamsa Balakrishnan. Modeling of aircraft takeoff weight using gaussian processes. *Journal of Air Transportation*, pages 1–10, 2018.

[28] Richard Alligier and David Gianazza. Learning aircraft operational factors to improve aircraft climb prediction: A large scale multi-airport study. *Transportation Research Part C: Emerging Technologies*, 96:72 – 95, 2018.

[29] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, pages 83–94, Berlin, Germany, April 2014.

[30] EUROCONTROL Experimental Centre. Coverage of 2016 european air traffic for the base of aircraft data (bada) versions 3.14 & 4.2. Technical report, EUROCONTROL, June 2017.

[31] Junzi Sun. World aircraft database [data file], 2017. Retrieved from http://junzis.com/adb/data.

[32] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[33] Christopher M Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[34] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[35] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[36] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[37] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Citeseer, 2013.

[38] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge-Volume 1526*, pages 40–51. CEUR-WS. org, 2015.

[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[40] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[42] Leslie N Smith. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE, 2017.

[43] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[44] Ira Shavitt and Eran Segal. Regularization learning networks: Deep learning for tabular datasets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1386–1396. Curran Associates, Inc., 2018.