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

Richard Alligier

1st International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT2020)

Predictive Joint 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. Focusing on the climb phase, we predict some of the unknown point-mass model parameters. These unknown parameters are the mass and the speed intent. This speed intent is parameterized by three values $(\mathrm{cas}_1, \mathrm{cas}_2, M)$. These missing parameters might be useful to predict the future trajectory of a climbing aircraft.**

In this work, an ensemble of neural networks uses the observed past trajectory of the considered aircraft as input and predicts a Gaussian Mixture Model (GMM) modeling the joint distribution of $(\mathrm{mass}, \mathrm{cas}_1, \mathrm{cas}_2, M)$.

Ideally, this predicted distribution will be close to a conditional distribution: the distribution of possible $(\mathrm{mass}, \mathrm{cas}_1, \mathrm{cas}_2, M)$ values given the observed past trajectory of the considered aircraft.

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 obtained data set contains millions of climbing segments from all over the world.

Using this data, we show that using the proposed predictive model instead of a regression model brings almost as much information as using a regression model instead of a simple mean.

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 [1], [2] envisioning trajectory-based operations. An accurate trajectory prediction is required for the new automated tools and algorithms implementing these concepts.

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 joint distribution of the mass m and the speed profile parameters $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$. More specifically, an ensemble of neural networks are trained to predict a Gaussian Mixture Model (GMM) modeling this joint distribution. These neural networks take the information available about the considered climbing aircraft as input and compute the predicted GMM.

The main contribution of this paper is to use machine learning on a large historical data set to predict the joint conditional distribution of the unknown parameters from the past points of a climbing aircraft.

The rest of the paper is organized as follows: Section I presents how the aircraft climbs, the object of this study. This might be useful for readers not familiar with aviation. Section II describes the considered trajectory prediction problem and Section III details how we model this problem as a machine learning problem. Section IV describes the data used in this study. Section V details the machine learning method used, and the results are shown and discussed in Section VI, before the conclusion.

I. How the Aircraft climbs

The climbing vertical profile of a flight can be very different from one to another. The aircraft climbs till it reaches its initial cruise altitude. This initial climb might contain level-off segments that stops the climb temporarily. These level-off segments are said to be inefficient regarding the fuel consumption and the noise annoyance compared with a continuous uninterrupted climb ([3]). These level-off segments might come from air traffic control operational restriction. These are the result of a balance between the required separation between the aircraft, the efficiency of the trajectories and the workload of both pilots and air traffic controllers.

If we only consider continuous climb segments, there is still a great variety of vertical profiles. The aircraft type has a great impact on the vertical performances: the maximum rate of climb of a small single-engine piston aircraft is way smaller than the rate of climb of a business jet. For the same aircraft, its weight may also vary from one flight to another depending on the fuel and payload. This impacts its vertical performance. The weather, especially the temperature ([4]) and the wind gradient ([5]), have also a great impact on the vertical performance.

A. How the Aircraft Climbs are Operated

A perfect knowledge of the flying object and weather are not enough as we consider a piloted flying object. We need to know how the aircraft is operated by the pilot. Roughly speaking, the pilot can act on the thrust setting and the pitch of the aircraft. The former controls the available power while the latter controls how this power is shared between the kinetic energy and the gravitational energy of the aircraft.

As the forces applied to the aircraft are functions of the airspeed, the pilot typically controls the pitch to follow a target airspeed profile.

Concerning the thrust setting, different strategies can be used by the pilot.

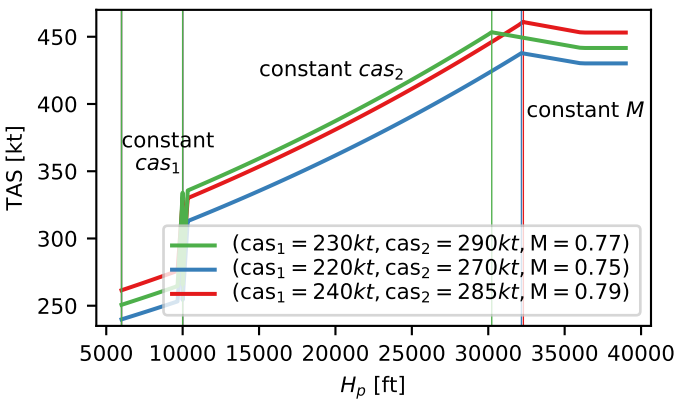

Figure 1: Three speed profiles parameterized with different values for $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$.

1) How the speed and Altitude are Quantified in Aviation: In aviation, three physical quantities are used to quantify the airspeed: the True AirSpeed (TAS) is the speed of the aircraft relatively to the air; the Calibrated AirSpeed (CAS) actually quantifies the dynamic pressure which is "converted" to the TAS required at the mean sea level to create the measured dynamic pressure ; and the Mach number is the ratio between the TAS and the speed of sound.

Likewise, the altitude usually used in aviation is not a distance between the aircraft and the ground. It is a pressure altitude usually denoted H_p. Roughly speaking, it is the measured atmospheric pressure "converted" to the altitude required to create the measured pressure according to a fixed atmosphere model: the ISA atmosphere model ([6]).

2) Typical Speed Profile Parameterized by $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$*:* The target speed profile is specified by two constant CAS segments followed by one constant Mach number segment. From the take-off, the aircraft accelerate till the CAS reaches cas_1. Then, the CAS is maintained equal to cas_1 till $H_p = 10000$ ft. Then, the aircraft accelerates till the CAS reaches cas_2. This CAS is maintained during the climb. The Mach number increases along a constant CAS climb. When this Mach number reaches M the aircraft switches to a constant Mach number climb. This happens at the *crossover altitude*. This altitude is a function of the values of cas_2 and M.

Figure 1 plots different speed profiles parameterized with different values $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$.

Although the vast majority of the flights follow such a speed profile, one can observe that some flights have a speed profile that does not comply with this parameterized speed profile.

3) Choice of the $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$ *Values:* The choice of the parameter $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$ of the speed profile is done to minimize the global cost. This cost includes the fuel costs and the costs related to the flight time. These time costs include the time related costs of the crew and maintenance of the aircraft ([7]). Of course, there is a trade-off between fuel cost and time cost: typically, if the aircraft goes faster, the time cost is reduced but the fuel cost increases. To specify the expected balance between the fuel costs and time costs, the pilot can enter the *cost index* CI inside the Flight Management System (FMS). This cost index "converts" the time cost to a fuel quantity of the same cost. Hence, the total cost to minimize is proportional to $\int_{t_0}^{t_f} FF(t) + CI dt$ where $FF(t)$ is the fuel flow ([8]). A $CI = 0$ will only minimize the fuel consumption. The larger the cost index is, the greater the $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$ selected values will be, and the lower the flight path angle will be ([7], [9]).

The cost index strategies can be different from one airline to an other ([7]). Furthermore, as the cost of the whole trip is optimized, the trip distance and the landing mass[1] have also an impact on the selected speed ([8]). As a consequence, one can observe a variety of $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$ values for the actual flights.

4) Thrust Setting: Aircraft are designed to meet a minimum climb performance in case of an engine failure at the maximum take-off weight (MTOW). As a consequence, in normal operation where all engines are properly running, one can operate takeoff and climb at a reduced thrust setting while still having a rate of climb at an acceptable level. The lower the mass is, the more the thrust can be reduced. Of course other factors might impact the thrust choice which includes the runway length, the runway contamination and the surrounding obstacles.

This thrust reduction can be implemented using a *derate* or an *assumed temperature*. Depending on the engine manufacturer, one or both methods are available. The *derate* method have a fixed number of possible thrust reduction. According this Rolls-Royce document [10], there are two thrust reductions available for the B777 and many other airplanes: 10 % and 20 %. In this document, from $H_p = 10000$ ft the thrust reduction is progressively reduced till the aircraft recovers its full max climb thrust. Depending on the selected taper strategy, the aircraft can recover its full thrust at $H_p = 12000$ ft or at $H_p = 30000$ ft. The second method, the *assumed temperature* method (also called FLEX), consists in entering in the FMS an Outside Air Temperature (OAT) higher than the actual OAT. This results in a lower thrust. According to regulations ([11]), this thrust reduction can not exceed 25%.

These reduced climb operations increase the climb distance and duration. It also increases the total fuel used for the trip. However, these reduced thrust climbs are expected to reduce the wear of the engines, reducing the maintenance costs ([10]).

All these different thrust setting strategies might contribute to a larger variety in the climb trajectories observed.

II. The Considered Trajectory Prediction Problem

This section describes the considered trajectory prediction problem.

A. Trajectory Prediction Problem

Considering a climbing aircraft, we want to predict its future climbing trajectory using only the information available in ground-based systems. We further assume that there are no level-segment in the future trajectory. In such systems, the mass, the speed profile and the thrust setting are not known.

[1]When minimizing the cost, the take-off mass is unknown as the fuel used for the trip is not known.

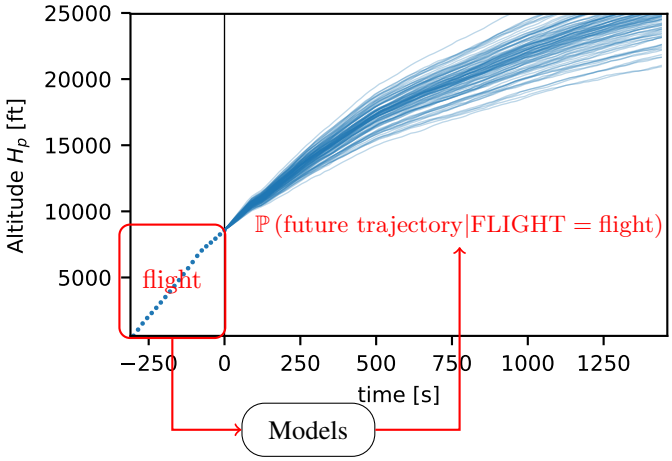

Figure 2: The trajectory prediction problem considered: using all the information we have about a climbing aircraft, we want to predict the conditional distribution of its climbing trajectory for the next minutes. This prediction is computed using models.

However, we have access to the aircraft type and variant, the past position and speed, and the flight plan data of the considered aircraft.

To handle the prediction uncertainty, we want to predict the probability distribution of the future trajectory, not just one predicted trajectory. Ideally, this predicted distribution will be the conditional distribution of the future trajectory given all the information we have about the considered flight. Figure 2 depicts the trajectory prediction problem considered.

B. Literature Review on Trajectory Prediction

There are typically two ways to obtain a trajectory prediction: 1) the kinetic approach which models the forces applied to the aircraft, allowing the computation of the motion using Newton's laws and 2) the kinematic approach which directly models the motion of the aircraft.

Among the works using the kinematic approach, few works have tried to handle the uncertainty of the future trajectory. Among these work, we can cite [12] that predicts intervals that contain, with a specified confidence, at least a desired proportion of the conditional distribution of the future altitude. We can also cite [13] which uses two Gaussian Mixture Models (GMM) to obtain one generative model of the sequence points for the departure and a second generative model for the arrival trajectories. These generative models can also be used to compute a probability distribution of the future trajectory by conditioning the GMM on the observed past positions. However, when we look at the conditional mean, this mean is a linear function of the past positions. This might limit the performance of this approach for trajectory prediction.

Among the works using the kinetic approach, some works ([14], [15], [16]) used past trajectory points to estimate the aircraft mass using a model of forces like the Eurocontrol Base of Aircraft DAta (BADA) model. 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.

Other kinetic approaches try to handle the uncertainty of the estimates. [17], [18] 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 [19], 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, [20] 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.

More recently, in [21] we have used a large historical ADS-B database to train an ensemble of neural networks. Each network predicts a Gaussian distribution. The networks in the ensemble are then combined to obtain the prediction of one Gaussian distribution. This method was applied to predict the mass and each parameter of the speed profile $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$. The predicted Gaussian have a diagonal covariance of size 4.

The current paper uses a similar approach. However, it predicts a GMM, not a diagonal covariance Gaussian.

III. Modeling the Considered Trajectory Prediction Problem as a Machine Learning Problem

This section describes how we model our trajectory prediction problem as a machine learning problem. The obtained machine learning problem is the one we try to solve in this paper.

Modelizations like the kinematic modelizations ([12], [13]) discussed in the previous section do not use a physical model at all. Our modelization heavily relies on one physical model, the Eurocontrol BADA model ([22]).

A. A Physical Model: BADA

BADA provides models of the forces applied to the aircraft. Used along Newton's laws, these forces' models can be used to compute the predicted motion of the considered aircraft. Equation 1 below is central in this model. It is obtained by considering the scalar product of the airspeed vector and the equation obtained with the second Newton's law.

$$V_a \frac{\mathrm{Thr} - \mathrm{Drag}}{\mathrm{mass}} = V_a \frac{dVa}{dt} + g_0 \frac{T}{T_{\mathrm{ISA}}} \frac{dH_p}{dt} \tag{1}$$

BADA models the thrust Thr and the drag Drag as functions of the altitude H_p, the airspeed V_a, the temperature T and the mass. T_{ISA} is the temperature at H_p in the ISA atmosphere.

In order to be used to compute the future trajectory, this physical model requires unknown parameters such as the mass, the thrust setting and the speed profile.

B. Thrust Setting and Mass

As seen in Sub-Section I-A4, there are different possible choices for the thrust setting.

In our work, we assume that the thrust setting is set to max climb thrust. We do not try to infer the thrust setting strategy.

With this assumption on the thrust setting, if we know the weather and the motion of the aircraft then the mass is the only missing parameter in Equation 1. Using this fact, in [15] we developed a method to extract the mass from the observed trajectory. Please note, that this method only relies on the second Newton's law, the BADA model and the thrust setting assumption. No assumption on the speed profile is required.

Because of the thrust setting assumption and the modeling error of BADA, this extracted mass is not the true mass. This extracted mass is the one minimizing the gap between the left-hand side and the right-hand side of Equation 1. Thus, this extracted mass mitigates the impact of the possible wrong assumption made concerning the thrust setting and the modeled forces.

C. Speed Profile

Concerning the speed profile, we assume that each aircraft follows a $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$ speed profile. This type of speed profile is described in Subsection I-A2. We consider that each aircraft will have its own $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$ values parameterizing its speed profile.

D. The Machine Learning Problem we Consider in this Paper

With the choices made in the two previous sub-sections, the required parameters to use the BADA model to compute a prediction are the mass and the $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$ values.

Using all the information we have about the considered climbing flight (denoted flight), we want to predict the probability distribution for these missing values $(\mathrm{mass}, \mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})$. For each situation, we want the predicted distribution to approximate the unknown conditional distribution $(\mathrm{mass}, \mathrm{cas}_1, \mathrm{cas}_2, \mathrm{Mach})\,|\mathrm{FLIGHT} = \mathrm{flight}$.

A previous work ([21]) predicts such a distribution. However, the method used in this previous work can only predict Gaussian distributions. Furthermore, it assumed that all the missing values are conditionally independent given FLIGHT.

In this paper, we also want to capture the fact that the missing values might not be independent: we want to predict the joint distribution of the missing values given FLIGHT.

IV. The Data: How It was Prepared

The trajectory data used in this study are from The OpenSky Network ([23]). The OpenSky Network is a participatory sensor network of ADS-B sensors that covers mainly Europe and North-America.

The weather comes from the Global Forecast System (GFS). 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.

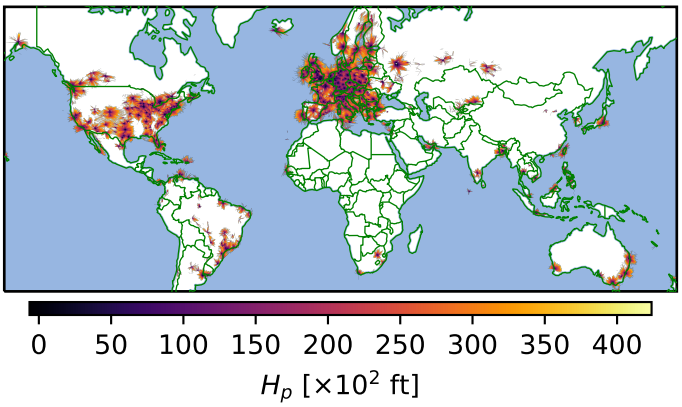

Figure 3: Climbing segments plotted on a world map. The mean altitude can be read from the color.

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. The description of this data set is more comprehensive in [24]. As described in [24], the preprocessing and sampling algorithms do not use future points to process the current point. This prevents any data leakage: no information from the future points are included in the current point.

Figure 3 plots these sampled climbing segments on a world map. The sample rate is one point every 15 s. In order to produce this figure, 331 millions aircraft positions were aggregated.

A. Splitting One Climbing Segment in Two Parts: Past and Future

We want to predict the future trajectory of a climbing aircraft from its past points. Thus, we must split each segment in two parts, the future part we want to predict and the past part that will be the input. This couple of past and future part is called *trajectory sample* in the rest of this section.

Knowing the current position and $p = 9$ consecutive past points, we want to predict the future q points. 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 4 illustrates two different trajectory samples (with $q = 40$) extracted from the same climbing segment.

B. Building One Example from One Trajectory Sample

Machine learning techniques use a set of (x, y) examples to build a model predicting y from x. 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. As discussed in III-B, the method used to extract the mass from these future points is the one described in [15]. This method assumes a max climb thrust.

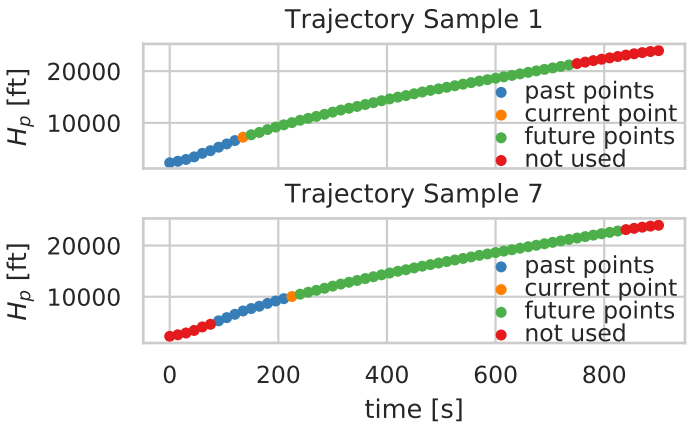

Figure 4: With $q = 40$, two different trajectory samples extracted from the same climbing segments.

2) Adding the Speed Profile: We want to extract cas_1, cas_2 and M 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 $(\mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$ minimizing the function e given by the equation (2). A method to efficiently minimize this error is given in [25].

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

V. Solving the Machine Learning Problem

We want to build a model predicting the conditional distribution $Y|X = x$ from a set $S = (x_i, y_i)_{1 \leqslant i \leqslant n}$ of n i.i.d. observations coming from the joint distribution (X, Y). In our case, x will be all the information we have about the considered flight whereas y will be the $(\mathrm{mass}, \mathrm{cas}_1, \mathrm{cas}_2, \mathrm{M})$ values.

The conditional distribution describes how the y values are spread out when $X = x$. This uncertainty is sometimes called *aleatoric uncertainty*. This uncertainty cannot be reduced. For instance, this uncertainty might come from a random variable that is not included in X but have an impact on the drawn y or more generally the inherent randomness of the process generating y. This uncertainty may vary depending on the considered x.

As the learned model will not be able to perfectly predict the conditional distribution, we want to include this prediction uncertainty inside the predicted distribution. This model related uncertainty is sometimes called *epistemic uncertainty*. This should result in having a predicted distribution wider than the actual conditional distribution.

A. Literature Review on Modeling Conditional Probability

In order to handle these uncertainties, some works rely on a Bayesian framework and others do not.

1) Works Explicitly Relying on a Bayesian Framework: Bayesian Neural Networks (BNNs) ([26]) are neural networks with random variables as weights w. Considering we have a *prior* probability $p(w)$ on these weights. If the *posterior* probability $p(w|S)$ is known, one can compute an approximation of the predictive distribution: $p(y|S, x) = \int p(y|S, x, w)p(w|S)dw$. The hard part is to obtain $p(w|S)$.

Samples from this posterior probability can be computed using Markov Chain Monte-Carlo methods ([26]). This approach is accurate but computationally expensive when considering a large set of observations. Other works ([27], [28]) use *variational inference* techniques to approximate $p(w|S)$ by $q_\theta(w)$ in a computationally efficient way. θ are parameterizing this approximating distribution $q_\theta(w)$. These techniques minimize a "distance", the reverse KL-divergence ([29]), between $p(w|S)$ and $q_\theta(w)$.

[30] argues that *dropout* ([31]) can be used to perform approximate Bayesian inference. They reported better results than [28], [27] on a set of regression problems. In [30], a deterministic neural network is trained using dropout layers. In contrast with the classic regularization use of dropout layers, these dropout layers are also active during the test phase. One evaluation of the network supposedly draws one sample from $p(y, w|S, x)$.

Gaussian Process Regression ([32]) is a powerful non-parametric framework that handles some sort of prior over random functions. As we consider a Gaussian Process, if we have a finite set of points $(x, x_1, \ldots, x_n)$ then $(f(x), f(x_1), \ldots, f(x_n))$ follows a joint Gaussian distribution. Its covariance matrix is given by the kernel function k : for each pair of inputs (z, z') we have $\mathrm{cov}(f(z), f(z')) = k(z, z')$. This kernel function k, chosen by the practitioner, can easily incorporate knowledge on the regression problem, like periodicity for instance. Using the joint Gaussian distribution, it is easy to condition the Gaussian to obtain $f(x)|f(x_1), \ldots, f(x_n)$. The naive exact computation of such a model requires $\mathcal{O}\left(n^3\right)$ operations which might be intractable for large data sets. Recent work ([33]) have reduced the complexity of exact inference to $\mathcal{O}\left(n^2\right)$. Other works ([34]) perform approximate inference by using approximations of the kernel matrix for instance.

2) Works that do not Explicitly Relies on a Bayesian Framework: In [35], the neural network directly predicts the mean and variance of $Y|X = x$. This network has two output vectors, one vector that shall predict $\mathbb{E}[Y|X = x]$ and an other vector 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 an ensemble of m networks, not a single neural network. These networks are obtained by using a different random initialization and different mini-batches with the same architecture and training set. These m networks are then combined to obtain the predicted mean and variance.

The idea of using a neural network to predict the parameters of a distribution modeling the conditional distribution is not a new idea. [36] proposed Mixture Density Network (MDN), a method using neural network to predict the parameters of a Gaussian Mixture Model (GMM).

If the considered input x is very unlikely in view of the

training data, it would be sound that the predicted uncertainty is high. However, for such an out-of-distribution x, all the neural networks methods discussed in this whole Section V-A might predict a low uncertainty ([37], [38]).

B. The Method we Used

In this work, using techniques similar to [35], [36], we train an ensemble of MDNs to predict one GMM. The MDNs inside the ensemble have the same architecture. They are trained with the same algorithm. However different RNG seeds are used. This will result in different initial weights and different mini-batches, as suggested in [35].

1) Gaussian Mixture Model (GMM): A Gaussian Mixture Model of K Gaussian models is described by a probability density function (pdf) that is a linear combination of K pdf of Gaussian models:

$$\mathrm{p}_{\alpha,\mu,\Sigma}(y) = \sum_{k=1}^{K} \alpha_k p_{\mu_k,\Sigma_k}(y) \tag{3}$$

In this above equation, the density function of each Gaussian model $\mathcal{N}(\mu_k, \Sigma_k)$ of dimension m is:

$$p_{\mu_k,\Sigma_k}(y) = \frac{\exp\left(-\frac{1}{2}(y-\mu_k)^t {\Sigma_k}^{-1}(y-\mu_k)\right)}{\sqrt{(2\pi)^m \det(\Sigma_k)}} \tag{4}$$

The mixture coefficients α_k must be positive and their sum must be equal to 1.

2) Computing the negative log-likelihood loss: Our neural network is trained by maximizing the log-likelihood of the training data. As a consequence, the *loss function* used to train our network will be the *negative log-likelihood* NLL.

We want to avoid the computation burden of inverting the covariance matrices. To do this, our network will predict the *precision matrix* $\Lambda_k \overset{\text{def.}}{=} \Sigma_k^{-1}$ and we will use the identity $\det(\Sigma_k) = (\det(\Lambda_k))^{-1}$ ([39]).

For each x, using one neural network, we can compute $\alpha_1(x), \ldots, \alpha_K(x)$, $\mu_1(x), \ldots, \mu_K(x)$ and $\Lambda_1(x), \ldots, \Lambda_K(x)$.

The negative log-likelihood of an example (x, y) is $\mathrm{NLL}(x, y) = -\log\ \mathrm{p}_{\alpha(x),\mu(x),\Sigma(x)}(y)$ which can be rewritten:

$$\begin{aligned}\mathrm{NLL}(x,y) &= -\log\left(\sum_{k=1}^{K}\alpha_k(x)p_{\mu_k(x),\Sigma_k(x)}(y)\right)\\ &= -\underset{k=1\ldots K}{\mathrm{logsumexp}}\left(\log\alpha_k(x) + \log\ p_{\mu_k(x),\Sigma_k(x)}(y)\right).\end{aligned} \tag{5}$$

The PyTorch implementation of the $\mathrm{logsumexp}$ operation use tricks to avoid overflow/underflow issues. Likewise, as $\alpha(x)$ is computed using a $\mathrm{softmax}$ operator, the $\log\alpha_k(x)$'s are computed using the $\mathrm{logsoftmax}$ operator which use similar tricks.

The log-likelihood of each Gaussian model is:

$$\begin{aligned}\log\ p_{\mu_k(x),\Sigma_k(x)}(y) = &-\frac{1}{2}(y-\mu_k(x))^t\Lambda_k(x)(y-\mu_k(x))\\ &-\frac{m}{2}\log(2\pi) + \frac{1}{2}\log\det\Lambda_k(x)\end{aligned} \tag{6}$$

Using the Equations (5) and (6), we can easily compute the NLL of an example (x, y) that was supposedly generated by the predicted GMM.

3) Computing the predicted GMM from the Output of a Neural Network: The predicted precision matrices must be symmetric positive definite. Such matrices can be rewritten as TT^t using the Cholesky decomposition where T is a lower triangular matrix. This decomposition is unique. Conversely, for every lower triangular matrix T with a positive diagonal, the product TT^t gives a symmetric positive definite matrix.

The GMM parameters are computed from the output of the neural network depicted in Figure 5. This output is the composed of tensors: $\log\alpha$ (size K), μ (size $K \times m$), L (size $K \times m \times m$) and d (size $K \times m$).

The L tensor is a collection of K lower triangular matrices with zeros on their diagonal. The d tensor stores the diagonals of these lower triangular matrices. These two tensors can be used to compute the Λ_k's precision matrices:

$$\begin{aligned}T_k(x) &= L_k(x) + \mathrm{Diag}(d_k(x))\\ \Lambda_k(x) &= T_k(x)T_k(x)^t\end{aligned} \tag{7}$$

The $\log\det\Lambda_k(x)$ term in Equation 6 can be easily computed using only the tensor d_k:

$$\log\det\Lambda_k(x) = 2\sum_{j=1}^{m}\log d_{k,j}(x) \tag{8}$$

Please note in Figure 5 that a softplus operation is used to be sure that all the computed $d_{k,j}$ are positive.

4) Data Preprocessing Before Optimization: The input variables are whitened using PCA. This process is a linear transformation, the coefficients of this linear operation are computed using only the training data. The output variables are standardized using only the training data. All these transformations will be applied on the validation data using the coefficients computed on the training data.

5) Weights Initialization and Optimization: Several optimization methods, learning rate schedules and activation functions were tested. Most of this tinkering to identify what worked best was done using the training and validation set, not the test set. During this phase, ranges of good values for the remaining hyper-parameters were identified. These are summarized in Table I.

At the end of this process we selected the AdamW ([40]) optimization method, the cyclical learning rate schedule ([41]) and the SELU activation function ([42]). The biases are left out of the weight decay regularization. The weight decay is handled inside the AdamW method. In this method, the weight decay differs from the L2 regularization.

As SELU was used, the weights of the linear operators in Figure 5 are initialized with a centered normal distribution with a standard deviation $1/\sqrt{fan\_in}$. The biases are initialized to 0.

The linear operators after "hidden layer n" on Figure 5 are initialized differently. The weights are initialized to zero. As a consequence, at first, the predicted GMM will be the same no

matter the input values. The predicted GMM is determined by the biases. These are initialized to have a "reasonable" guess for this predicted GMM.

As the output data is standardized, we initialize the biases to have: identity covariances for the Gaussians inside the GMM; same mixture coefficients. In order to break symmetry, the mean of the Gaussians are however initialized randomly from a centered normal distribution of standard deviation 0.5.

6) Hyper-parameter Search: The hyper-parameter search is done using random search. The hyper-parameters are drawn from the distributions in the Table I. The network is trained using all these hyper-parameters and the selected one is the one with the lowest NLL on the validation set.

The learning rate is not considered as an hyper-parameter here. An automatic procedure described in [41] is used to determine the initial learning rate.

The architecture of the fully connected neural network is an hyper-parameter. The number of hidden layers h is a random integer between 1 and 10. For each hidden layer the number of unit is drawn with replacement from a set of integers. The obtained unit numbers are then sorted in order to have many units at the first layer and less units at the last layer.

For each aircraft type, one hundred random sets of hyper-parameters were tested. The hyper-parameters with the best performance on the validation set were selected.

Table I: A summary of the distribution used to perform the random search of the hyper-parameters.

hyper-parameter description	distribution
number the hidden layers h	$\mathcal{U}([\![1, 10]\!])$
units of one layer	$\mathcal{U}(\{10, 20, ..., 100\} \cup \{200, 300, \ldots, 700\})$
dropout rate	$\mathcal{U}([0, 0.9])$
embeddings dimension	$\mathcal{U}([\![1, 10]\!])$
weight decay	$10^{\mathcal{U}([-3,0])}$
batch size	$\mathcal{U}(\{512, 1024\})$
number of components K	$5 * \mathcal{U}([\![1, 4]\!])$

7) Using an Ensemble of Neural Networks: Once the hyper-parameter is selected, an ensemble of E neural networks are then trained independently using a different RNG seed as suggested in [35]. Each neural network predicts a GMM. To combine these predictions, we consider a mixture of these predicted GMMs with uniform weights. As a consequence, the predicted pdf is just the average of the pdf of the E predicted GMMs:

$$p_{\text{ensemble}}(y) = \sum_{e=1}^{E} \frac{1}{E} \sum_{k=1}^{K} \alpha_k^e(x) p_{\mu_k^e(x), \Sigma_k^e(x)}(y) \tag{9}$$

Please note that this mixture of GMMs is just a GMM with $K * E$ components.

Roughly speaking, we hope that the K components of each GMM will model the *aleatoric uncertainty* whereas the combination of E GMMs will handle the *epistemic uncertainty*.

VI. Experiment Setup and Results

All the statistics presented in this section have been computed on the *test set* S_{Test}, data that has not been used in the model building process.

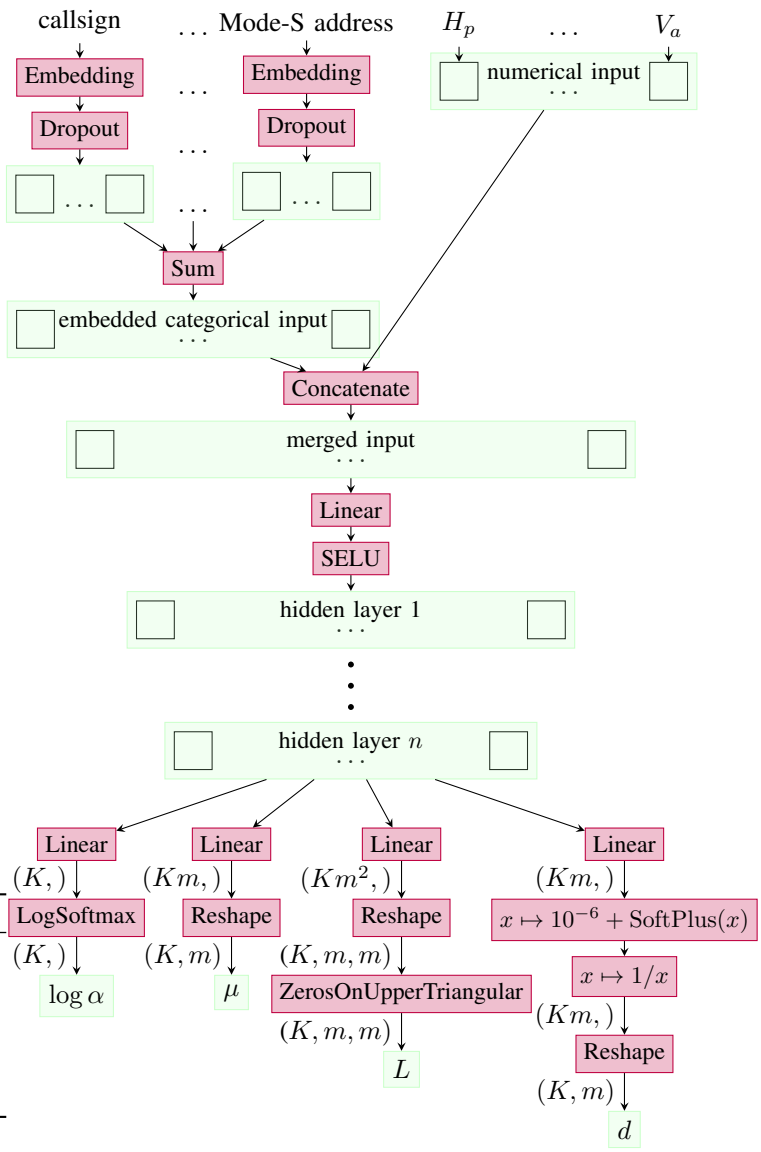

Figure 5: Architecture of the neural network we used. The manipulated vectors are in green whereas the operations applied to these vectors are in red. The input is at the top, the output is at the bottom.

A. Which Data Sets are Used to Fit and Evaluate our Models

Figure 6 illustrates how the data of the year 2017 has been used in this study.

The ten first months were used to build the predictive models. The last two months were used to evaluate the models performance.

During the model building process, the first ten months are split in two: the *training set* S_{Train} with the first eight months and the *validation set* S_{Valid} with the September and October months. These two sets are used to perform the hyper-parameter search; the training set S_{Train} is used to fit models with various hyper-parameter and the validation set S_{Valid} is used to compute the mean NLL of the fitted models. The hyper-parameter with the lowest mean NLL is selected. Then using this hyper-parameter, the final model is fitted on the first

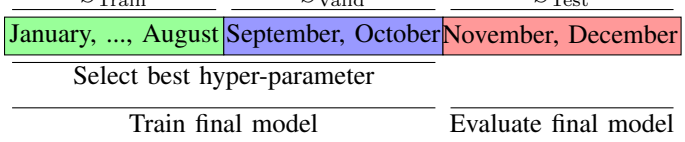

Figure 6: This figure illustrates how the data was split into sets and how these data sets are used to train and evaluate our models.

ten months.

This final model is then evaluated on the two remaining months, the test set S_{Test}.

As a matter of fact, a simple hold-out validation is used to select the hyper-parameter and to evaluate the final model.

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.

B. Which Variables are Used

We want to predict the distribution of $Y = (\text{mass}, \text{cas}_1, \text{cas}_2, \text{Mach})$ given $X = x$.

The x is a vector of all the information we have about the considered flight. More specifically, x contains all the variables described in Table II. All these variables are computed using the ten points between $t_0 - 9 * 15s$ and t_0, weather data and flight plan data.

The past positions and speeds are also used to compute an estimated mass and energy variations. These quantities are also included in x as they are related to the mass we want to predict.

For each example, we also have categorical variables such as 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.

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 $H_p = 0$ at current latitude and longitude	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

C. Considered Models

Different models are compared. Except for $\text{GB}(x)_{\text{Diag}}$ and $\text{GB}(x)_{\text{Full}}$, all these models are built using only the first ten months: the data sets S_{Train} and S_{Valid}. These models are listed in order of increasing complexity:

- Diag: This model predicts a multivariate Gaussian distribution with a diagonal covariance. This Gaussian $\mathcal{N}(\mu, \text{Diag})$ has the same mean and covariance for all the x. Hence, the predicted distribution is independent from x;
- Full: Same as Diag but with a full covariance;
- $\text{GB}(x)_{\text{Diag}}$: The model predicts a Gaussian $\mathcal{N}(GB(x), \text{Diag})$ where GB is a gradient boosted trees model predicting the mean and previously tested in [24]. The predicted covariance is constant, independent from x. The covariance $Diag$ is fitted using the error made by the gradient boosted model. As the error on the training set is much lower than the test set one, we chose to fit the error on the test set S_{Test}. Thus, this model is a "specific mean, constant covariance" model that only serves as a basis of comparison;
- $\text{GB}(x)_{\text{Full}}$: Same as $\text{GB}(x)_{\text{Diag}}$ but the covariance is a full covariance.
- $\text{Diag}(x)$: This model uses an ensemble of neural networks as described in Section V. Each neural network predicts a multivariate Gaussian distribution with a diagonal covariance $\mathcal{N}(\mu(x), \text{Diag}(x))$. Each network has an architecture similar to the one in Figure 5 with $K = 1$. The L computed by the network is not used, we consider $L = 0$. This model is similar to the one tested in [21];
- $\text{Full}(x)$: Same as $\text{Diag}(x)$ but the computed L is used. As a consequence, the predicted distribution by each network is a multivariate Gaussian distribution with a full covariance.
- $\text{GMM}(x)$: Same as $\text{Full}(x)$ but with $K > 1$. As a consequence, each network predicts a GMM.

D. Empirical Average Negative Log-Likelihood

Probability density functions (pdf's) are sensitive to scaling[2]. Our pdf aggregates variables that have a very different range of values. The mass is orders of magnitude larger than Mach. As a consequence, each y variable is centered and scaled using the mean and variance computed on the first ten months.

Table III presents the average NLL computed on the S_{Test} for different models. We want the observed y to be very likely according our predicted distribution. Thus, the lower the average NLL is, the better our model is. From an information theory perspective, this average NLL is the empirical cross-entropy between the true distribution and the predicted distribution.

Diag has an average NLL close to the entropy of a unit variance Gaussian[3]. This was somewhat expected because the y variables were standardized using the training set.

The Full model does not greatly reduces NLL except for the DH8D. For the other aircraft types, the full covariance fitted on the y's are almost diagonal. The largest term for all these covariances is between cas_2 and Mach. However, except for DH8D, this term is always inferior to 0.44.

For all the predictive models, the general observation is that the "full" version only reduces slightly the NLL over the "diag" version. This is somewhat surprising because the associated covariances do not model the same thing. For the Full model, it models Y; for the $\text{GB}(x)_{\text{Full}}$, it models the errors $y - GB(x)$ over the whole training set; and for $\text{Full}(x)$, it supposedly models $Y|X = x$. None of these different problems seem to significantly benefit from a full covariance model.

Going from Diag to $\text{GB}_{\text{Diag}}(x)$ gives a large 2.2 NLL reduction on average. The former is a model for which the predicted Gaussian distribution is independent from x. For the latter model, only the standard deviation of the predicted Gaussian is independent from x. The mean is predicted using the gradient boosted tree regression method. Roughly speaking, this NLL reduction quantifies the information gained when we use the Gradient Boosted regression method instead of a simple mean.

Going from $\text{GB}_{\text{Full}}(x)$ to $\text{Diag}(x)$ gives a large 1.15 NLL reduction. With the latter, the predicted diagonal covariances are specific to the considered x.

The last big NLL reduction observed is between $\text{Full}(x)$ and $\text{GMM}(x)$ with a 0.66 reduction. With the latter, the predicted distributions are GMMs and might be very different from a Gaussian.

In the end, using $\text{GMM}(x)$ instead of $\text{GB}_{\text{Diag}}(x)$ provides a 2.0 NLL reduction. This has to be compared with the 2.2 NLL reduction observed when using $\text{GB}_{\text{Diag}}(x)$ instead of Diag. Roughly speaking, the information gained by using our approach instead of a regression method is nearly as large as the one gained by using a regression method instead of a simple mean.

[2]For instance, $\forall \lambda \neq 1$, $\forall y \neq 0$, $p_{\mathcal{N}(\mu,\sigma)}(\mu + y) \neq p_{\mathcal{N}(\mu,\lambda\sigma)}(\mu + \lambda y)$.

[3]$\mathbb{E}_{Y\sim\mathcal{N}(0,I_4)}[-\log p_{\mathcal{N}(0,I_4)}(Y)] = 5.68$.

Table III: For each x, the model predicts a probability density function (pdf) $y \mapsto p_{\text{Model}}(y|x)$. Using these pdf's, we can compute the average Negative Log-Likelihood of the observed y's: $\underset{(x,y)\in S_{\text{Test}}}{\text{mean}} [-\log\ p_{\text{Model}}(y|x)]$. The lower this value is, the better the performance is. Intuitively, we want the observed y to be very likely according our predicted distribution.

model	Diag	Full	$\text{GB}(x)_{\text{Diag}}$	$\text{GB}(x)_{\text{Full}}$	$\text{Diag}(x)$	$\text{Full}(x)$	$\text{GMM}(x)$
A319	5.82	5.70	3.74	3.65	2.37	2.26	1.46
A320	5.87	5.73	3.56	3.50	2.09	2.03	1.24
A321	5.88	5.75	3.56	3.51	2.29	2.22	1.49
A332	5.79	5.59	3.37	3.33	2.07	2.01	1.20
B737	5.77	5.66	3.58	3.52	2.26	2.23	1.61
B738	5.92	5.80	3.67	3.62	2.13	2.07	1.49
B77W	5.56	5.44	3.13	3.09	1.68	1.62	0.85
CRJ9	5.96	5.80	3.79	3.74	2.85	2.79	2.20
DH8D	5.89	5.19	3.06	2.74	2.13	1.73	1.36
E190	5.72	5.56	3.95	3.86	2.99	2.90	2.16
E195	5.65	5.49	3.91	3.81	2.93	2.85	2.19

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 will be available at the author's GitHub page. 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 have trained an ensemble of neural networks to predict a Gaussian Mixture Model (GMM). Considering a climbing aircraft, the predicted GMM supposedly models the conditional probability $p\left(\text{mass}, \text{cas}_1, \text{cas}_2, \text{Mach}|\text{FLIGHT} = \text{flight}\right)$ where flight contains 10 past points of the flight, flight plan data and weather data.

If this model is good, the likelihood of the observed values according to our predicted distribution should be high. Using this principle, the Negative Log-Likelihood (NLL) is used to score the considered models.

In addition to the "GMM" model, other models were tested. These models are increasingly complex. This gives us a series of NLL scores that allows us to put into perspective the NLL score obtained with the"GMM" model.

Among these models, the "Diag" model does not use any flight information and always predicts the same distribution. Another model, "GB_{Diag}", is based on a regression model ([24]). It is used to compute the mean of the predicted distribution whereas the variance of this distribution is assumed to be constant. The flight information is only used to predict the mean, not the variance.

We observe almost the same NLL difference between "Diag" and "GB_{Diag}" than between "GB_{Diag}" and "GMM". Roughly speaking, using "GMM" instead of a regression model brings almost as much information as using a regression model instead of a simple mean.

Despite these results, neural networks might be fragile, especially when out-of-distribution inputs are used. In that respect, Gaussian Process Regression models with a good prior might be more reliable.

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.

REFERENCES

[1] S. Consortium, "Milestone Deliverable D3: The ATM Target Concept," Tech. Rep., 2007.

[2] H. Swenson, R. Barhydt, and M. Landis, "Next Generation Air Transportation System (NGATS) Air Traffic Management (ATM)-Airspace Project," National Aeronautics and Space Administration, Tech. Rep., 2006.

[3] ICAO, "Continuous climb operations (cco) manual," 2013.

[4] A. Gerretsen and S. Swierstra, "Sensitivity of aircraft performance to variability of input data," *EUROCONTROL Doc. CoE-TP-02005*, 2003.

[5] F. Huchet, "Introduction à la prévision de trajectoire," 2006.

[6] D. Poles, "Revision of atmosphere model in bada aircraft performance model," EUROCONTROL, Tech. Rep., 2010.

[7] Airbus, "Getting to grips with the cost index," Blagnac,France, 1998.

[8] R. Dalmau and X. Prats, "Fuel and time savings by flying continuous cruise climbs: Estimating the benefit pools for maximum range operations," *Transportation Research Part D: Transport and Environment*, vol. 35, pp. 62 – 71, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361920914001825

[9] D. Poles, A. Nuic, and V. Mouillet, "Advanced aircraft performance modeling for atm: Analysis of bada model capabilities," in *Proceedings of the 29th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Oct 2010.

[10] W. James, P. O'Dell, and R. Royce, "Derated climb performance in large civil aircraft," in *Rolls–Royce, Conference Boeing Performance and Flight Operations Engineering*, 2005.

[11] FAA, "Ac 25-13 - reduced and derated takeoff thrust (power) procedures," 1988.

[12] M. G. Hamed, R. Alligier, and D. Gianazza, "High confidence intervals applied to aircraft altitude prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2515–2527, Sep. 2016.

[13] S. T. Barratt, M. J. Kochenderfer, and S. P. Boyd, "Learning probabilistic trajectory models of aircraft in terminal airspace from position data," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[14] Y. S. Park and D. P. Thipphavong, "Performance of an Adaptive Trajectory Prediction Algorithm for Climbing Aircraft," in *2013 Aviation Technology, Integration, and Operations Conference*. 08, Aug 2013. [Online]. Available: http://dx.doi.org/10.2514/6.2013-4263

[15] R. Alligier, D. Gianazza, M. G. Hamed, and N. Durand, "Comparison of Two Ground-based Mass Estimation Methods on Real Data (regular paper)," in *International Conference on Research in Air Transportation (ICRAT), Istambul*, may 2014.

[16] M. Uzun and E. 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*, vol. 18, no. 2, pp. 323–345, 2017.

[17] J. Sun, J. Ellerbroek, and J. Hoekstra, "Bayesian inference of aircraft initial mass," in *Proceedings of the 12th USA/Europe Air Traffic Management Research and Development Seminar*. FAA/EUROCONTROL, 2017.

[18] J. Sun, J. Ellerbroek, and J. M. Hoekstra, "Aircraft initial mass estimation using bayesian inference method," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 59 – 73, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X18302626

[19] J. Sun, H. A. Blom, J. Ellerbroek, and J. M. Hoekstra, "Aircraft mass and thrust estimation using recursive bayesian method," p. (on line), June 2018. [Online]. Available: www.icrat.org

[20] Y. S. Chati and H. Balakrishnan, "Modeling of aircraft takeoff weight using gaussian processes," *Journal of Air Transportation*, pp. 1–10, 2018.

[21] R. Alligier, "Predictive Distribution of the Mass and Speed Profile to Improve Aircraft Climb Prediction," in *ATM Seminar 2019, 13th USA/Europe ATM R&D Seminar*, Vienna, Austria, Jun. 2019. [Online]. Available: https://hal-enac.archives-ouvertes.fr/hal-02138151

[22] V. Mouillet, "User manual for base of aircraft data (bada) rev.3.14." EUROCONTROL, Tech. Rep., 2017.

[23] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. 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*, ser. IPSN '14, Berlin, Germany, April 2014, pp. 83–94.

[24] R. Alligier and D. Gianazza, "Learning aircraft operational factors to improve aircraft climb prediction: A large scale multi-airport study," *Transportation Research Part C: Emerging Technologies*, vol. 96, pp. 72 – 95, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X18311896

[25] R. Alligier, D. Gianazza, and N. Durand, "Machine learning applied to airspeed prediction during climb," in *Proceedings of the 11th USA/Europe Air Traffic Management R & D Seminar*, Lisbon, Portugal, 2015.

[26] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[27] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International Conference on Machine Learning*, 2015, pp. 1861–1869.

[28] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, 2011, pp. 2348–2356.

[29] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[30] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[33] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018, pp. 7576–7586.

[34] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *arXiv preprint arXiv:1807.01065*, 2018.

[35] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

[36] C. M. Bishop, "Mixture density networks," 1994.

[37] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *arXiv preprint arXiv:1906.02530*, 2019.

[38] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[39] G. Dorta, S. Vicente, L. Agapito, N. D. Campbell, and I. Simpson, "Structured uncertainty prediction networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5477–5485.

[40] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[41] L. N. Smith, "Cyclical learning rates for training neural networks," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 464–472.

[42] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in neural information processing systems*, 2017, pp. 971–980.