Flight Level Prediction with a Deep Feedforward Network

Matthias Poppe, Roland Scharff, Jörg Buxbaum
DFS Deutsche Flugsicherung GmbH
Langen Germany
Matthias.poppe@dfs.de

Debora Fieberg
University of Heidelberg
Heidelberg, Germany
d.fieberg@stud.uni-heidelberg.de

***Abstract* – A deep feedforward network has been used to predict the flight level with look ahead time up to six minutes for climbing flights. Representing features were developed which provide adequate flight characteristics. Mode S Enhanced Surveillance data from more than 400,000 real world flights were collected to calculate the feature vector. After supervised training of the network, the results for different prediction horizons will be presented.**

Keywords - Air Traffic Control, trajectory prediction, machine learning, feedforward neural network

I. INTRODUCTION

In Air Traffic Control, current and future operational concepts rely on the accurate prediction of the aircraft trajectory [1]. Controller Decision Support tools like Conflict Detection or Conflict Resolution tools require a good knowledge about the future position of an aircraft in lateral, vertical and longitudinal dimension. Of particular interest is the improvement of the vertical part of the trajectory, which seems to be one of the most challenging research questions because of the high bandwidth of vertical climb rates, speed development and the associated operational and technical uncertainty [2]. The prediction is already rather accurate in the lateral plane because aircraft follow their route with a high precision, defined by the Required Navigation Performance (RNP).

A successful and more accurate vertical prediction would be highly advantageous for the Controller Assistance Tools (CATO), which provides conflict detection and resolution support for Air Traffic Controllers [3]. Since these tools do not use aircraft performance models like *Base of Aircraft Data* BADA [4], the predicted flight level is calculated as follows: Whenever two planes are foreseen to come closer than six Nautical Miles (NM) on the lateral axis, a vertical conflict calculation is kicked off. A prediction for the next six minutes (value derived from SESAR real time validations [5][28]) will allow the controller in lower airspace to successfully resolve potential conflicts in advance.

For now, this is being done by simply taking the most recent rate of climb from the radar with a static buffer of ±500 feet per minute. This buffer is needed in order to take into account the vertical uncertainty and to obtain a good compromise between false and missed alarms [5]. However, this procedure necessarily results in a higher workload for controllers due to the likely increase in alarm rates. With more potential conflicts to resolve, their attention is constantly captured inducing stress and perhaps even mistrust in the system. To reduce false alarms while keeping the rate of missed conflicts reasonably low, a more accurate prediction would be beneficial. The miss rate or false negative rate cannot be quantified at this stage because it depends on the working methods of the Air Traffic Controller.

The focus of this work lies in predicting an aircraft's altitude immediately after Take Off up to Flight Level 285 (1FL = 100ft) after its initial climb by making use of machine learning methods with supervised learning in form of a deep feedforward neural network.

In the past, many research activities regarding trajectory prediction (TP) rely on parametric models where typical aircraft characteristics are derived from the EUROCONTROL BADA model [4]. [6] provides an extensive overview of TP research studies. The majority uses a point-mass model with the problem of estimating the a priori not known input data, e.g. Take Off weight. Another unknown is the operational environment of the flight, which may impose other constraints that heavily impact the performance of the aircraft [7].

Current research tries to overcome part of these problems by applying machine learning techniques to estimate some of the most important parameters, e.g. the Calibrated Airspeed [8] or the aircraft mass at Take Off [9]. By training a network with the observed trajectory data, the BADA parameters could be adjusted accordingly and the future altitude could be predicted with a significantly reduced root mean square error. In [10], a hybrid system based on BADA has been developed in an online context where known altitudes were used to fit the model parameters and to predict the remaining part of the trajectory for the climbing phase. This paper again confirms the need to use aircraft derived data to feed the BADA model.

Non-parametric approaches are less frequent because in general they require a huge amount of input data for sufficient training. In [11], a single hidden layer feed forward network was trained with a relatively small set of trajectories. It has been trained for a selected aircraft type and already demonstrates the

potential of these networks. Other approaches [12] use genetic programming in order to learn the structure of the variables of a linear regression. Disadvantage of the non-parametric trajectory prediction is the difficulty to generalize the model once it has been trained, and, as mentioned before, the required data base.

In this paper, the main contribution stems from the fact that recorded data from real operations for more than three months with hundred thousands of flights from all major German airports were available. They were used for selecting features, training and validation of a neural network in order to predict the flight level of the next six (configurable parameter) minutes. The advantage with this approach is the good generalization because the results are not constrained to a specific aircraft type or operational ATC/airport environment. It is assumed that the Feedforward network is able to 'learn' and differentiate the operational environments.

After a general introduction to neural networks, the chosen feature vectors for the Machine Learning approach as well as their extraction from the available Mode S data will be explained. Section IV provides some feature statistics and first results on a large data set. The outlook includes some ideas how to combine this approach with other machine learning techniques.

II. NEURAL NETWORKS

A. *Brief Introduction*

In general, three different ways of Machine Learning can be distinguished: supervised learning, unsupervised learning and reinforcement learning. A thorough introduction to the subject can be found in [13].

The main objective for supervised learning, which is supposed to be applied for this trajectory prediction task, is to train a model with labeled training data. This trained model allows to make predictions of future yet unknown input data. The training data are often called 'features' x of an observed phenomenon (the explanatory variables) while the output data are called 'labels' y. We assume that there exists a relation $y = f(x)$ between x and a target variable y. In general, a model h is learned with examples of the outputs $(y_0, \dots, y_N)$ associated with the inputs $(x_0, \dots, x_N)$, approximating f. A performance measure P will be applied to measure the abilities of the machine learning algorithm.

Supervised learning itself can be broadly divided into two different main categories: classification and regression. In classification tasks, the output y belongs to a category or group. For example, we could forecast the aircraft type given certain input trajectories to train a network. In regression tasks, y is a variable with a 'real' numerical value or a vector of values. In our problem, we will use regression techniques to forecast the flight level in up to six minutes, expressed as a numerical value, e.g. FL326.

Unsupervised learning is used to discover structures in big data sets and to extract useful information without knowing the corresponding output y like in supervised learning. Unsupervised learning problems can be further divided into association and clustering problems. An association rule learning problem is where you want to discover rules that describe large portions of your data. A clustering problem is where you want to discover the inherent groupings in the data. In Air Traffic Control, for example, it could be interesting to group aircraft with similar performance without knowing certain parameters like the Cost Index or the Take Off weight.

In Reinforcement Learning, a computer program will interact with a dynamic environment in which it must perform a particular goal (such as playing a game with an opponent or driving a car). The program is provided with feedback in terms of rewards and punishments as it navigates through its problem space. Being exposed to this environment of continuous training, the machine learns to make specific decisions with the aid of its algorithm. For example, an Arrival Manager could learn to turn and space the aircraft with minimum separation while receiving feedback in terms of the number of landings per hour (the more the better) in relation to other parameters.

B. *Feedforward Netwok*

Figure *1* shows a feedforward network with one input layer $\boldsymbol{x}$, two hidden layers $\boldsymbol{h}^i, \boldsymbol{h}^j$ and one output layer $\boldsymbol{h}^l$.

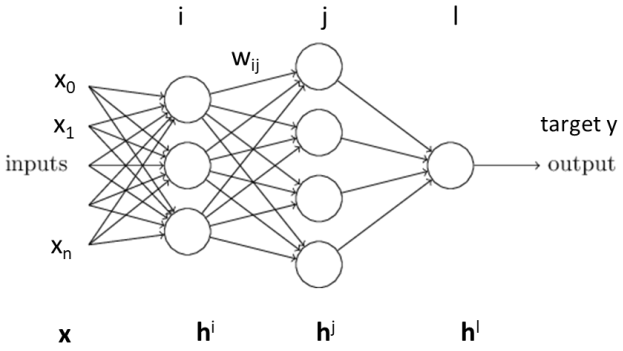

Figure 1. Schematic diagram of a feedforward network

The feedforward network defines the mapping of the supervised target $\boldsymbol{y} = h(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the value of the parameters θ that result in the best function approximation. $\boldsymbol{\theta}$ may consist of weights w and a bias b. In this example, layer j computes an output vector $\boldsymbol{h^j}$ using the output $\boldsymbol{h^i}$ of the previous layer, starting with the input $\boldsymbol{x} = \boldsymbol{h^0}$. Thus:

$$\boldsymbol{h^j} = \sigma(\boldsymbol{b^j} + W^j \boldsymbol{h^i}) \qquad (1)$$

with W^j as a matrix of weights and σ as the in general non-linear activation function. The rectified activation function is the default function recommended for use with most feedforward neural networks [13]:

$$\sigma(u) = \begin{cases} 0, & u < 0 \\ u, & u \geq 0 \end{cases} \qquad (2)$$

The goal is to minimize the cost function or the loss of our model. In most cases, our parametric model defines a distribution $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ and we can use the principle of maximum likelihood, i.e. we use $-\log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ as cost function. Other cost functions can be used as well, e.g. predicting the median value of $\boldsymbol{y}$ for each $\boldsymbol{x}$ (mean absolute error). In optimizing the cost function, it is expected that the chosen performance measure P is optimized as well.

In general, the hidden layers (i, j, see Figure *1*) in a well trained neural network form a "good" representation of the data, which helps to make good predictions. Multiple layer neural networks with deep architectures are more difficult to train than those with shallow architectures [14]. With random initialization, the layers closer to the input layer are poorly trained. Much better results can e.g. be achieved with unsupervised pre-training. It is anyway a big challenge to optimize all hyperparameters of a deep feedforward network.

III. Data Set and Used Features

We have restricted the input data to easily available radar data that could be obtained from the vast majority of aircraft. Since there is a Mode EHS mandate in Germany since March 2005 for all aircraft with a maximum take off mass in excess of 5.7 tons [15], [16], the current equipage rate for IFR flights is more than 98% in Germany (according to DFS internal statistics from 2017, *Lage- und Informationszentrum*).

It is essential that the raw input data are pre-processed in order to obtain a representative feature vector $\boldsymbol{x} \in \boldsymbol{R^n}$. A well chosen set of features can greatly enhance the predictive capabilities of the chosen machine learning model. For this reason, we have put some effort in defining and preprocessing the raw input data in order to obtain features that are well suited for this task.

A. Extraction and Decoding of Mode-S EHS

Selected Mode S Enhanced Surveillance Data (EHS) were collected in spring and summer 2018 in order to extract the required information for data preparation. Seven radar stations all over Germany were considered:

- Frankfurt South FFS/ASR
- München South MUS/ASR
- Düsseldorf DUS/ASR
- Nordholz NHZ/SREM
- Tegel TGL/ASR
- Auersberg AUB/SREM
- Gosheim GOS/SREM.

ASR: Airport Surface Radar
SREM: Surveillance Radar Equipment Medium-Range

The date and time are converted into a timestamp and the CAT48 Item 250 Mode-S Enhanced Surveillance (Mode-S EHS) data were decoded [17]. The Mode-S EHS data contain the Comm-B Data Selector (BDS) number with the available register. Further details of the BDS register can be obtained from [18]. Selected were:

1. BDS 4.0 - Selected Vertical Intention
2. BDS 5.0 - Track and Turn Report
3. BDS 6.0 - Heading and Speed Report

The flight example in Figure 2 shows the different extracted parameters and how they evolve over time. The IAS restriction of 250 knots (red line) below FL100 at about 200 seconds can be seen. After that, the acceleration takes place with the typical drop in the vertical rate (orange line). Then, the IAS remains constant until the change to Mach at ≈ 780 seconds. The Flight Control Unit (FCU) Selected Altitude (blue line) reflects the current clearance. The figure also shows a typical initial peak in the climb rate shortly after take off (≈ 3000 feet/minute).

The two vertical bars mark the prediction interval: the earliest predictions are made 60 seconds after take-off while the latest predictions are made not later than Flight Level 220. The main reason for this choice is to avoid making forecasts, once the aircraft has been leveled off.

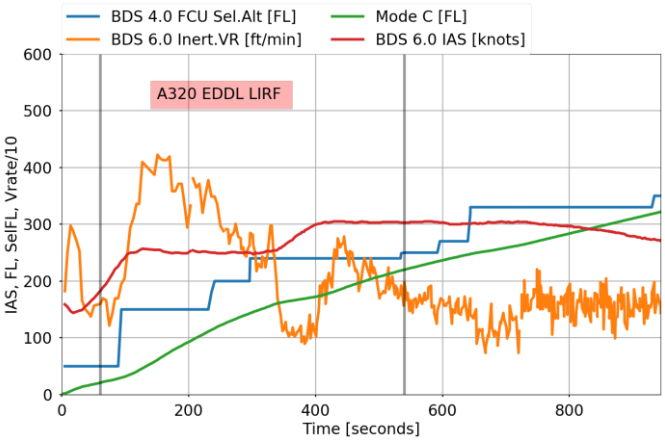

Figure 2. Example of obtained Mode S flight data

B. Data Cleansing

Being interested in making predictions for climbing flights, a flight was defined to be "climbing" whenever the recorded data fulfills:

$$min_t\{FL(t)\} < 20,\ max_t\{FL(t)\} > 285 \quad \text{and}$$
$$argmin_t\{FL(t)\} < argmax_t\{FL(t)\} \qquad (3)$$

The data are filtered such that these conditions hold true. In addition, all datasets for which no time stamp was available were dropped since no information can be gained from these. Unavailable call signs were replaced by a numeric ID. A posteriori, it was observed that a few data sets contained a call

sign of question marks or blank spaces. These flights were also discarded.

Furthermore, the radar stations were not always capable of capturing all data so that we had to deal with incomplete datasets. We used the benefits of the *Python numpy* library [19] as well as interpolation techniques for this purpose. In total, more than 400.000 climbing flights all over Germany were extracted using this cleansing method.

C. Feature Extraction

In order to determine which features are predictive for the dynamics of the climbing phase, information from multiple sources were gathered, including the knowledge of controllers, pilots and Airbus employees, statistics of the given datasets, and a full flight simulator study. All features are calculated using the Mode S EHS data and the aircraft type.

The motivation for the chosen features was to keep the Machine Learning network relatively small with fewer layers and less neurons per layer, compared to an unfiltered set of features with less predictive characteristics. The training is also expected to be faster and more efficient.

Ultimately, the following features have been selected:

a) *Take Off Safety Speed V_2 (x_0)*

We want to get an approximation of the V_2 take off safety speed, which is the minimal speed an airplane needs to climb to the first safe altitude with one engine inoperative. Typically, all engines will be used, so usually the aircraft will fly with $V_2 + 10$ knots. This speed will be reached shortly after takeoff (see Figure 3). When the acceleration altitude between 1000-2000 feet above ground is reached, the aircraft will accelerate to the next maximum allowed airspeed.

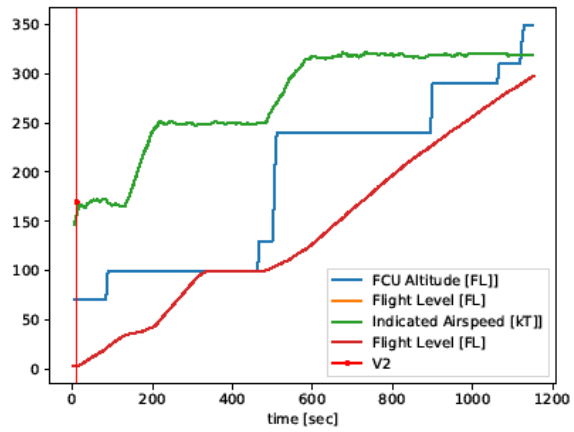

Figure 3. Take Off safety speed V_2 at 0 seconds (red dot at IAS)

The data suggested to use the first finite entry of the IAS, where the flight level is less than 1000ft for mimicking the V_2.

Thus, the feature was set to be

$$V_2 \coloneqq IAS(t_x)\ with\ t_x\ such\ that\ FL(t_x) \leq 1000 \quad (4)$$

As the V_2 speed is related to aircraft-specific parameters such as weight, temperature, and wind, it is expected to gain some information about aircraft performance characteristics. It varies considerably across aircraft types [20][21].

b) *Average and Actual Rate of Climb (x_1, x_2)*

This feature describes the average and actual value of the rate of climb (ROC) over time. From the data, three different rates of climb were extracted: the inertial vertical velocity, the barometric altitude rate and the rate of climb calculated from the flight level and time. Since numeric differentiation is noisy and may lead to huge errors, data from Mode-S were used. Due to the measurement method, the inertial vertical velocity is more accurate than the barometric altitude rate. For all features, the inertial ROC is used if more than 80% of the data points are available, and the one with more available data points in any other case.

c) *ROC Peak below 2800ft (x_3)*

Controllers observed that during the initial climb phase, there is a vertical speed peak immediately after takeoff, which - to a certain degree - seems to correspond with the rate of climb after FL100 when the IAS becomes constant. Indeed, a moderate correlation of this peak and the average rate of climb between FL100 and FL285 exists (correlation coefficient r = 0.40).

The peak might be typical for the performance of different aircraft types and could give a hint to the energy share factor used depending on the cost index of a particular flight. The maximum climb rate value below 2800ft was used as a value for this peak for our feature vector:

$$x_3 \coloneqq \max\{ROC(t) \mid FL(t) < 2800\} \quad (5)$$

The value of 2800ft was chosen, but could be replaced by other altitudes in this range. It is important to include the characteristic peak, but not any other peaks that might occur later in time due to the acceleration phase.

d) *Wind below FL285 (x_4, x_5)*

The wind component in flight direction influences the aircrafts performance during climb as well . The average wind in flight direction is calculated from the ground speed (GS), the true air speed (TAS), the true track angle as well as the magnetic heading h. The heading has to be adjusted to the geographic/true heading h_{true} by adding the angle between magnetic north and geographic north for Germany (VAR).

$$h_{true} = h(t) + VAR \quad (6)$$

For the sake of simplicity, a constant declination of $VAR = 2.0$ degree is assumed. Compared to other errors and the resolution of the BDS register, this assumption may be justifiable. The wind components in flight direction are binary coded (0: tailwind, 1: headwind). A second bit indicates the wind strength above a certain threshold. This threshold has been set to

40 knots. Only some wind data are actually finite since there are missing data sets involved in the calculations. The main cause of this problem is the sparse availability of the BDS 5.0 register. The selected radars (except München) receive this register only every 10th antenna revelation.

e) Heading Change Flag (x_6)

It is assumed that the influence of our wind feature (x4, x5) would be less predictive if the aircraft makes bigger turns. Therefore, we include a feature of heading changes, which is the accumulation of observed heading changes. A flag is used to indicate whether these changes are above a certain threshold. Currently, this threshold has been set to 160 degree.

f) Altitude Restriction Flag (x_7)

Especially below FL100, the aircraft is restricted in its climb rate by the Flight Control Unit (FCU) or Control and Display Unit (CDU) settings because of the trade-off between acceleration to constant CAS and vertical rate. We assume that restrictions will not change very much the average climb rate, but that there might be some deviations arising due to restrictions like level-off clearances.

Therefore, a Boolean flag was set, which is true, if the aircraft comes closer than a certain threshold to the FCU selected altitude. It is assumed that this impacts the average climb rate although we have observed that aircraft tend to compensate this level-off period by higher climb rates later on (of course, within the limits of aircraft performance envelopes).

$$x_7 := \begin{cases} 0, & if\ \exists\, t: FCU(t) - 100 * FL(t) \leq \omega \\ 1, & else \end{cases} \quad (7)$$

After discussion with Air Traffic Controller, ω := 1400 [feet] was chosen.

g) Time since FL100 (x_8)

Furthermore, the time that passed since Take Off (in seconds) was included. The work of Maximilian Beierl [22] has shown, that climb rates can well be fit by an exponential model after FL100 with $ROC(t) = a * e^{-bt}$. It might be useful for the network to include the following feature:

$$x_8 = t - t_{100} \text{ where } FL_{t100} = 100 \quad (8)$$

h) Last FL (x_9 to x_{13})

To provide a history and the development over time of the flight level, five features were included:

- FL 60s ago … FL 0s ago (actual FL).

To obtain these values, the FL needs to be interpolated to estimate the values if they were missing at these particular times. Calculations of the linear interpolation include all previous flight levels since 60 seconds ago.

i) Last IAS (x_{14} to x_{18})

For the same reason, an interpolation of the IAS at the above mentioned times is also included:

- IAS 60s ago … IAS 0s ago (actual IAS).

If the aircraft is not yet flying with constant IAS, this feature may well describe the acceleration (or in general: the change) of the speed.

j) Aircraft Type (x_{19})

The aircraft type has been added as a separate feature since this study was carried out with all aircraft types.

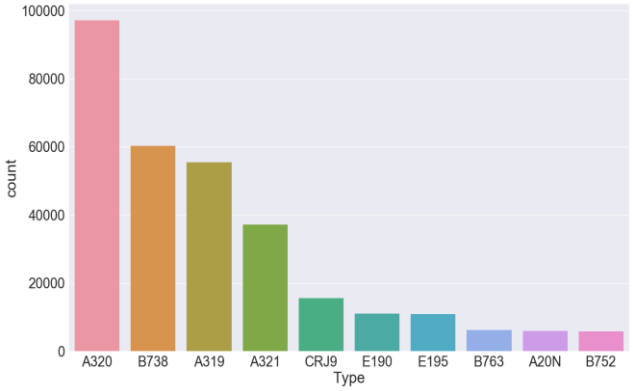

Figure 4. Most frequent aircraft types

. In summary, more than 120 different types, ranging from business jet to military aircraft, have been observed *Figure 4* shows the ten most frequently collected aircraft types. The types have been encoded using a) label encoding and b) one hot encoding. In both cases, no significant difference in the overall results could be observed.

IV. Feature Statistics

In order to check whether the extraction yield reasonable features, the distribution of the features for all flights was analysed. Figure 5 to Figure 7 show the distribution for some selected features.

A. Distributions

The altitude restriction flag (Figure 5) indicates that more than $2/3$ of all aircraft could climb without restriction in the sense that the difference between FCU Selected Flight Level and actual Flight Level is always more than 1400 feet.

The average rate of climb (Figure *6*) can be fitted with a Gaussian distribution. We observed a mean of about 2216 feet per minute (fpm) and a standard deviation of 438 fpm. The distribution of IAS (Figure *7*) interestingly shows two characteristic peaks. This may be due to 250 knots restriction

below FL100 and due to the constant airspeed regime, which aircraft apply after the acceleration phase during climb. In the future, one could perhaps come up with different criteria or additional clustering methods in order to sort aircraft into different performance groups, e.g. according to aircraft types.

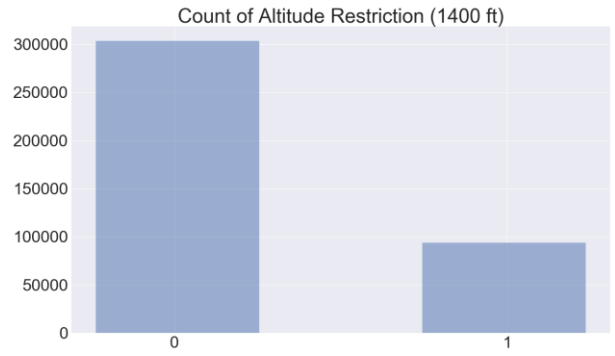

Figure 5. Altitude restriction flag

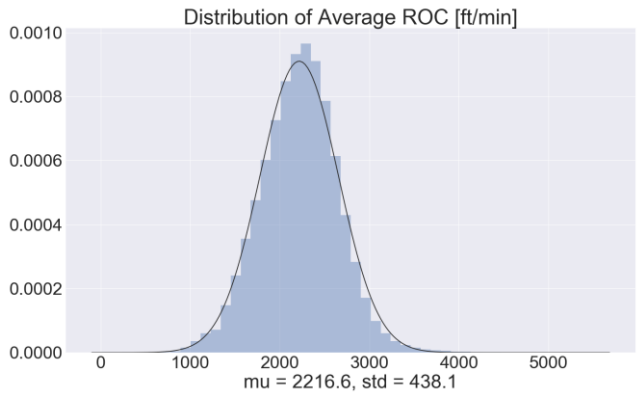

Figure 6. Average Rate of Climb (ROC) assuming a normal distribution

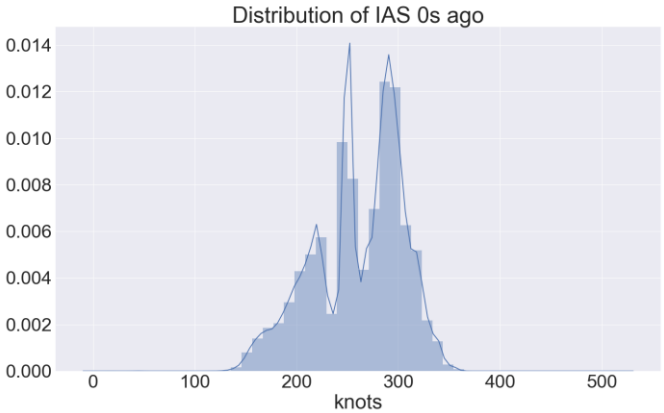

Figure 7. Distribution of current IAS

B. *Label Creation, Splitting and Scaling*

The aim is to predict the flight level of the aircraft for the next six minutes (360s) as this is the currently configured look ahead time for CATO conflict detection. It was decided to label each feature vector with a four-dimensional vector containing the FL values in

$$n * 90 \ \ seconds \qquad | \, n = \{1,2,3,4\} \qquad (9)$$

Again, to obtain these values, an interpolation of the flight level had to be used. Due to performance considerations, the network was trained separately for each FL prediction horizon.

For each flight, random indices were chosen and the feature vector as well as the corresponding labels were created for each index (only the data up to this index are provided, as it will be the case in future applications). In total, close to 400.000 features and label were generated leading to a feature vector of shape $\boldsymbol{X}(397.242, 20)$. The data were randomly split with shuffling into 80% training data and 20% test data.

In a second step, we standardized the features and labels by removing the mean and scaled them to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples. Standardization of a dataset is a common requirement for many estimators: they might behave badly if the individual feature do not look like standard normally distributed data. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected [23].

V. MACHINE LEARNING RESULTS

The neural network has been developed with *Keras*, a well known deep learning library, together with *tensorflow* [24][25]. These open source software libraries support flexible machine learning architectures and allow computation across a variety of platforms including Graphic Processor Units (GPU).

The obtained results were achieved with a network of six hidden layers while each layer contains a different number of nodes in descending order with one node for the output layer. The rectified linear unit has been chosen as activation (3) for each layer. Different optimizers were tested (ADAM, Adadelta, stochastic gradient descent SGD, refer to [26] for an overview) but no significant differences in convergence could be observed. *Figure 8* shows an example of the learning curve for training and validation (test) data with SGD.

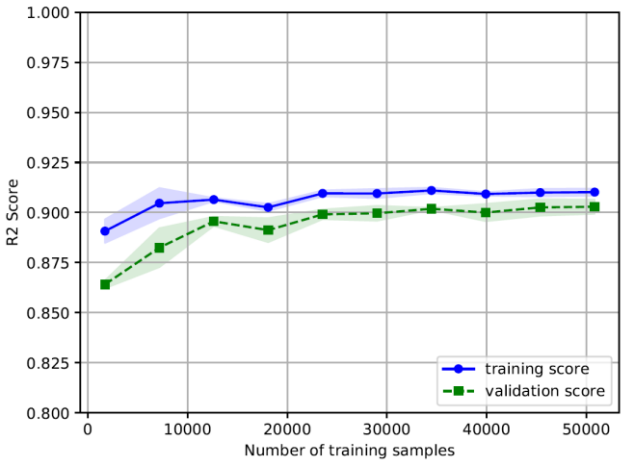

Figure 8. Training and validation loss

While the model fit with the training data further improves with the number of epochs, the loss of the test data seems to improve only quite slowly. Further evaluations with the number of layers and neurons per layer may lead to smaller gaps between training and test data and thus a better bias-variance trade-off. For assessment of the model performance, a k-fold cross validation (CV) has been used. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds. The scorer function (SF e.g. mean squared error) is then computed and the procedure is repeated k times.

$$CV = \frac{1}{k} \sum_{i=1}^{k} SF_i \qquad (10)$$

The R^2 statistic provides a measure of fit. It measures the proportion of variability in Y that can be explained using X. An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression [27]. For $k = 8$ and using the R^2 statistic as scorer function, the resulting mean score and standard deviation is 0.922 ± 0.002 (360 seconds prediction horizon).

After training of the network with the training data set, the output predictions are generated with the previously splitted test data. We can now compare the predicted flight level with the features (i.e. the 'true' flight level) and perform an evaluation with the mean and standard deviation of the error between prediction and test data. *Figure 9* shows the resulting distributions for 360 seconds prediction horizon. At first glance, there seems to be a good match although the network seems to predict higher levels in the first segment. This can probably be explained with the acceleration phase of the aircraft at the beginning of the flight that results in a significant drop of the climb rate. It seems that this drop in climb rate is not taken into account to full extent.

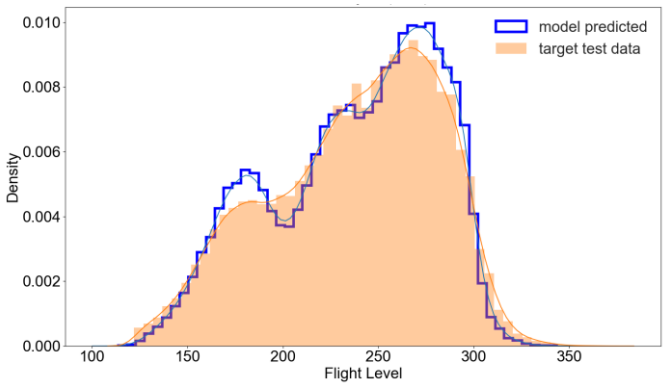

Figure 9. Distribution of predicted flight level (test data)

TABLE I. provides for different prediction times the obtained R^2score, the mean absolute error and the standard deviation of the error, considering the absolute difference between network predicted flight level and the measured flight level. The number in brackets indicate the values training the network only with A320 aircraft (and suppressing feature x_{19} *Aircraft type*).

As expected, all metrics improve with a shorter prediction horizon. With the chosen network architecture, there is only a minor difference between the training and the test data, which indicates no problems with overfitting. Assuming a standard vertical separation of 1000 feet or 10 FL, we observe a mean error of about 8 FL for a prediction horizon of six minutes. This means that the mean error is less than the vertical separation standard. The standard deviation reflects the uncertainty, to some extent caused by human (controller and pilot) intervention. If the *Managed Mode* flag in the Mode S EHS could be made available, the standard deviation could possibly be decreased. Up to now, only few modern aircraft (e.g. B787, A350) transmit this flag.

TABLE I. PREDICTION METRICS ALL FLIGHTS (A320)

Metric	**Prediction Time 360 seconds**	
	Test Data	***Train Data***
R^2 score	0.922 (0.931)	0.931 (0.942)
Mean error	8.14 (7.58) FL	7.63 (6.78) FL
Std. Dev. error	7.21 (6.84) FL	7.04 (6.56) FL

Metric	**Prediction Time 270 seconds**	
	Test Data	***Train Data***
R^2 score	0.953 (0.956)	0.959 (0.963)
Mean error	7.01 (6.68) FL	6.49 (6.02) FL
Std. Dev. error	6.49 (6.30) FL	6.27 (6.01) FL

Metric	**Prediction Time 180 seconds**	
	Test Data	***Train Data***
R^2 score	0.975 (0.978)	0.978 (0.981)

Metric	Prediction Time 180 seconds	
	Test Data	*Train Data*
Mean error	5.35 (5.13) FL	4.95 (4.53) FL
Std. Dev. error	5.32 (5.16) FL	5.13 (4.85) FL

The results for a single selected aircraft type (here: A320) are slightly better than for all aircraft types together. This may be due to less variance in the performance parameters although we had less training (77.000) and test (19.000) data available. With a perfectly trained network and One Hot encoding for the aircraft type, we could finally expect the same mean error.

VI. CONCLUSION AND FUTURE WORK

The obtained results show that the mean error for the flight level prediction horizon of up to six minutes is much better than the currently chosen static buffer in CATO of ± 500 feet per minute, which results in ± 30 FL in six minutes. Given the high operational and also technical uncertainty in climb behavior of aircraft, the results seem to be promising. The cross validation with the high number of training and test data confirms the validity of the results, without any limitations with regard to the aircraft type. The developed features describe the general climb behavior and allow to train the neural network, obtaining in all cases statistical R^2scores above 0.9. The authors conclude that this approach allows to reduce the uncertainty buffers for Decision Support Tools in the climb phase, which is the most challenging part of a flight in terms of trajectory predictions.

Future work should extend this neural network to look ahead not only 6 minutes but e.g. in 30 seconds intervals. The number of layers and neurons could also be increased to improve the overall performance. Regularization like Early Stopping or Dropout may also enhance the results and prevent overfitting. Additional benefit would be to include the controller vertical rate clearances, which reflects the human intervention. These clearances could be made available via the ATM system, Data Link or Voice Recognition. The feature importance should be studied with the objective to reduce the number of features, e.g. with *Sequential Backward Selection* algorithm. It is also envisaged to cluster the aircraft types with similar performance according to their climb behavior, which reduces the size of the respective feature vector. The authors also believe that a combination of this neural network with recurrent neural networks like Long Short-Term Memory (LSTM) sequence models will further improve the performance because the dynamics of the early aircraft acceleration phase could be taken into account for the later climb behavior.

REFERENCES

[1] Christian Bousmanne et.al., SESAR 2020 Concept of Operations Edition 2017, EUROCONTROL, November 2017

[2] Common Trajectory Prediction Capability for Decision Support Tools, Sip Swierstra, EUROCONTROL HQ, Brussels, Steven M. Green, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, CA, 5th USA-Europe ATM Seminar, Budapest, June 2003

[3] Controller Assistance Tools Requirements Specification, Development Requirements - Release 6 / Final for industrial Prototype, Version 1.0, 22.01.2015

[4] A. Nuic. User Manual for the Base of Aircraft Data (BADA), Revision 3.12, EUROCONTROL, August 2014

[5] Poppe, M., Herr, S., Reinhardt, K. & Achatz, G. (2015). Erste Validierungsergebnisse der Controller Assistance Tools an der iCAS-SESAR-Plattform, Innovation im Fokus 2/15

[6] Ben Musialek, Carmen F. Munafo, Ryan Hollis, and Paglione Mike. Literature Survey of Trajectory Predictor Technology. Technical Report DOT/FAA/TCTN11/1, Federal Aviation Administration, William J. Hughes Technical Center, 2010.

[7] Martin Lindner. Entwicklung eines Algorithmus zur Anpassung der Vorhersage von Trajektorien an spezifische Flug-manöver von Luftverkehrsgesellschaften. Technical University of Dresden, 2013

[8] Richard Alligier, David Gianazza, Nicolas Durand. Machine Learning applied to Airspeed Prediction during Climb. ATM Seminar 2015, 11th USA/EUROPE Air Traffic Management Seminar

[9] Richard Alligier, David Gianazza, Nicolas Durand. Ground based estimation of Aircraft Mass, adaptive vs. least square method. ATM Seminar 2013, 10th USA/EUROPE Air Traffic Management Seminar

[10] Areski Hadjaz, Gaetan Marceau, Pierre Saveant, Marc Schoenauer. Online Learning for Ground Trajectory Prediction. SESAR 2nd Innovation Days, 2012.

[11] Yann Le Fablec and Jean Marc Alliot. Using Neural Networks to predict aircraft trajectories. In Proceedings of te International Conference on Artificial Intelligence (ICAI 99), Las Vegas, 1999

[12] Caikun Zhang; Xuan Zhang; Chuan Shi; Wenfu Liu. Aircraft Trajectory Prediction based on genetic programming. 3rd International Conference on Information Science and Control Engineering (ICISCE), July 2016

[13] Ian Goodfelow, Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press Cambridge, Massachusetts, London, England, 2016

[14] Yoshua Bengio. Learning Deep Architectures for AI. Foundations and Trends in Machine Learning. Vol. 2, No. 1. 2009

[15] Honeywell. White Paper Mode S and the Europeamn Mandates. Revision 13, 2004

[16] DFS Deutsche Flugsicherung GmbH. Aeronautical Information Circular IFR 7, SSR Mode S Enhanced Surveillance. January 2003

[17] Debora Fieberg . Feature Preparation for a Machine Learning Prediction of Trajectories of Climbing Flights, Voluntary Internship at DFS. 2018

[18] ICAO Doc 9871 AN/464 Technical Provisions for Mode S Services and Extended Squitter, First Edition, 2008

[19] www.numpy.org

[20] Dr Darren Rhodes, Capt. Spencer Norton. Aircraft Climb Performance. Presenation 2017

[21] Airbus Flight Operations. Getting to grips with Aircraft Performance. January 2002

[22] Maximilian Beierl, OTH Regensburg, March 2018. 'Trajektorienprädiktion'. Arbeit zur Erlangung des akademischen Grades Bachelor of Science

[23] Scikit-learn developers. Scikit-learn User Guide. Release 0.21.dev0. 2018

[24] Francois Collet et.al. Keras. 2015. https://keras.io

[25] Martin Abadi et.al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org

[26] Sebastian Ruder. An overview of gradient descent optimization algorithms. Insight Centre for Data Analysis, NUI Galway. June 2017

[27] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning. Springer New York Heidelberg Dordrecht London. 2013

[28] SESAR Project 04.07.02 Deliverable D09, Validation Report Exercise VP175, Edition 00.01.01, September 2015