Predicting flight routes with a Deep Neural Network in the operational Air Traffic Flow and Capacity Management system

Herbert Naessens, Thomas Philip, Marcin Piatek, Kristof Schippers, Robert Parys
17 December 2017
EUROCONTROL Maastricht Upper Area Control Centre
Horsterweg 11, NL-6199 AC Maastricht Airport, The Netherlands
www.eurocontrol.int/muac/

Herbert.Naessens@eurocontrol.int

Abstract

Trajectory prediction is an essential component of Air Traffic Management systems but is hampered by route uncertainty because of air traffic controller clearances in the tactical phase. By augmenting traditional trajectory prediction logic with machine learning, a considerable improvement to accuracy can be achieved.

A deep neural network is trained on historical trajectories and a set of predictors. The neural network predicts the most likely route through the airspace, and has some ability to generalise to flights and conditions not seen before. By iterative training on newly recorded data, the neural network can keep up with changes.

The neural network has been integrated in the operational Air Traffic Flow and Capacity Management system of the EUROCONTROL Maastricht Upper Area Control centre. By the use of 'What-If' trajectories, the new approach enhances existing capabilities rather than replacing them.

Keywords

Trajectory prediction, neural network, machine learning, Air Traffic Flow and Capacity Management, route uncertainty

Introduction

Flight trajectory prediction underpins much of the functionality of air traffic management systems, both in the tactical (air traffic control) and pre-tactical (air traffic flow & capacity management) phases of a flight. Trajectory prediction is the process of estimating the future states of individual aircraft based on the current aircraft state, estimation of pilot and controller intent, expected environmental conditions, and computer models of aircraft performance and procedures. Systems in use today generally apply predefined rules and models to predict trajectories from available input data. Prediction logic is static and is grounded on domain knowledge of human experts and kinematic equations.

Volatility and the intrinsic uncertainty of input data, as well as aspects affecting actual aircraft trajectories that are difficult to model, reduce the accuracy of the predicted trajectories. Deviations are often substantial, degrading performance of the ATM system.

Two elements in particular have been found problematic at the EUROCONTROL Maastricht Upper Area Control Centre:

- Uncertainty of departure times.
- Flights not conforming to the route in the filed flight plan because air traffic controllers give permission to fly shorter routes. The instructions originate from controllers in the local centre or colleagues in upstream centres. They are driven by a multitude of factors and may change over time.

This paper describes a method to address route uncertainty by using machine learning. A deep neural network is trained on historical trajectories and a set of predictors. The neural network predicts the most likely route through the Maastricht UAC airspace, and has some ability to generalise to flights and conditions not seen before. By iterative training on newly recorded data, the neural network can keep up with changes.

The paper also describes how the neural network has been integrated into the existing trajectory prediction logic in the operational system. By the use of 'What-If' trajectories, the new approach enhances existing capabilities rather than replacing them. This way, strengths are combined and a path to gradually increase the role of machine learning is created.

The paper concludes by measuring performance of the integrated solution.

Illustration of the problem

Figure 1 illustrates a typical case where the predicted trajectory deviates from the actual flown trajectory because of instructions by air traffic controllers.

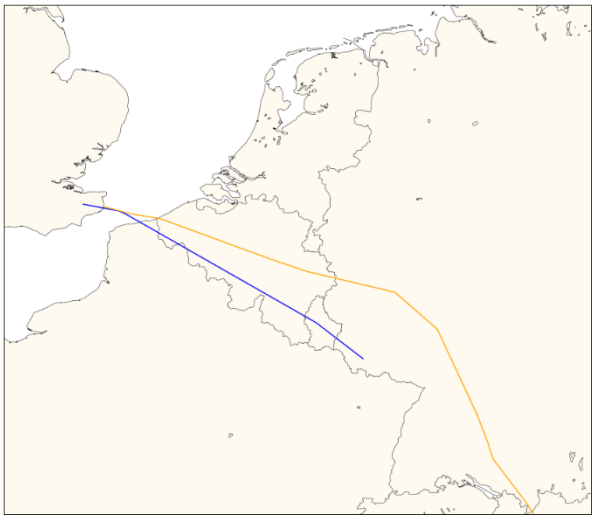

Figure 1: filed versus flown route

The line in yellow shows the initial predicted trajectory of a flight from London to Rome. The initial prediction is derived from the flight plan filed by the pilot[1]. The pilot intends to enter Maastricht UAC airspace via waypoint 'KOK'. The route continues to the east, avoiding military areas in the south of Belgium, and leaves Maastricht UAC airspace via waypoint 'MATUG'. Eventually it turns southward above Germany. This route is based entirely on fixed waypoints and routes.

The line in blue shows the route actually flown by the aircraft. Only the part relevant to Maastricht UAC is depicted. The flight enters Maastricht UAC airspace slightly to the south of waypoint KOK and is allowed to fly a direct route to a waypoint in the south of Germany. Different sectors are crossed which invalidates workload planning. The deviating route decreases the accuracy of predicted hotspots and medium term conflicts. Because crossing times and exit point are different, there is also an impact for the downstream control centre.

Current trajectory prediction logic takes into account controller instructions, and will eventually adapt the initial prediction. However, this is only possible at the time the instruction is entered into the system or when a consequential event is detected; e.g. a lateral deviation between radar plots and the predicted route or a change to transfer conditions for an incoming flight.

Being able to predict these deviations a sufficient time upfront would bring great benefits to trajectory prediction. Unfortunately, the factors that drive air traffic controller decisions are complex and intertwined. Moreover, they depend on working procedures or habits that may change over time.

Because it is hard to express concise rules describing actual flown routes, other approaches have been studied that build on recent advances with machine learning. The growing amount of recorded historical data is a key enabler.

Rationale for a deep neural network

Several machine learning methods have been evaluated, including Decision Trees, Random Forests, Kernel SVMs, K-Nearest Neighbours and Neural Networks. A random forest with adequate pruning offered the best results out of the box. However, after research and careful tuning, a deep neural network could surpass the results by a small margin. The decision to continue with a neural network is mainly driven by:

[1] The route in yellow is extracted from the EFD messages received from the European Aviation Network Manager.

- Random forests do not scale as well if more training data or more predictors are used.
- The serialised model is much smaller for a neural network.
- Off-the-shelf libraries offer a high degree of customisability, e.g. to try custom cost functions or experiment with alternative topologies, and allow integration with existing application code.

Training data, validation data and test data

Because the concept of machine-learned trajectories is fundamentally different from the existing approach, the new functionality is initially restricted to a subset of the traffic. This allows supervisors and flow managers to grasp the trajectory differences and get acquainted with the concept of predicting trajectories from past data rather than from the route filed by the pilot. The selected traffic covers flights from the UK to European destinations to the south and southeast. The traffic amounts to about 10% of all Maastricht UAC traffic and has been selected on the basis that it suffers heavily from route deviations.

Flight and airspace data, including actual flown trajectories observed from radar tracks, was taken from the period 15 December 2015 – 4 July 2017, minus some busy days. 246.993 flights are included in the dataset. A random subset of 5% of the data was kept for model selection and hyper-parameter tuning (cross-validation). The other data was used to train the neural network.

Performance of the neural network was tested on unseen data collected in the period 23 August 2017 – 10 October 2017 and busy days left out from the training data.

Results mentioned in this paper apply to the initial operational implementation. Note however that a neural network has also been trained for all Maastricht UAC traffic with positive results.

Target data to be predicted

The data to be predicted is the horizontal part of the flight trajectory (the route). Several operations are performed on the observed flown trajectories in the training set prior to training the neural network.

1. An intersection is made between the observed trajectory and the Maastricht UAC Area of Responsibility (AoR) volume. The latitude/longitude coordinates of the AoR route are converted to x/y geodetic coordinates relative to the AoR centre point.
   The horizontal route is then simplified to 4 points by iteratively applying the Douglas-Peucker algorithm. The resulting simplification is sufficiently close to the flown trajectories: For 99.6% of the flights, the lateral deviation does not exceed 5NM at any point along the trajectory. For 89%, the lateral deviation does not exceed 1NM.

   The habit of issuing clearances to fly direct routes to waypoints is a likely explanation.
   For some trajectories, the simplification to 4 points results in collocated points. To avoid artefacts with the ordering of points in the output of the neural network later on, collocated points are spaced evenly.

2. The x/y coordinates of the 4 points are rotated and scaled. The line from the filed entry point (NCOP) to the filed after-boundary exit point (BPXCOP) becomes the new X-axis. Origin is at the NCOP. The coordinates on the X-axis are scaled by 0.5. The coordination transformation serves 3 purposes:

- It serves as a normalization for the target data. The orientation of the new X-axis corresponds approximately to the overall forward direction for the flight. The new Y-axis represents the lateral deviation, i.e. turns to the left or right. Performance of the neural network was slightly better after applying this normalisation. Some rare cases with extreme errors were avoided.
- Scaling along the new X-axis allows for a more optimal cost function. The neural network cost function is explained in detail later on. One of the elements is the distance of the first and last point of the prediction versus the first and last point of the observed trajectory. Errors along the longitudinal direction of the flight are less costly than errors perpendicular to it.
- The transformation allows for generic sanity checking on the output data, e.g. ordering of the points or smoothing of extreme angles.

It should be noted that the transformation is based on points that are also used as input data to the neural network and that are known upfront to the prediction.

Predictors

Input to the neural network is:

- Entry coordination point (NCOP), Exit Coordination point (XCOP), After-Boundary Exit point (BPXXCOP), Entry flight level (NFL), Requested flight level (RFL) and Exit flight level (XFL) retrieved from the initial flight plan, which itself is based on the data filed by the pilot/airline. The BPXCOP is the waypoint after the boundary with the downstream air traffic control centre.
- Coordinates of the destination airport
- Compass bearing between the NCOP and the departure airport
- Day of the week
- The half hour interval the flight is expected to enter the AoR. This is taken from the first estimate available.
- Reservation of military areas. Reservation data for various military areas is received from multiple sources and consolidated into a single schedule. To avoid dependencies on names, the AoR is divided in square grid cells measuring 15NM x 15NM. Military activity, expressed as the upper reserved FL, is indicated per cell. Rationale is that names and geographical definitions may change over time and that names may not be consistent across systems. The grid cell mapping ensures that, in case geographic definitions change, old training data still reflects useful information. The approach also supports the future use of weather data as a predictor. Knowledge about the upper reserved FL, in combination with NFL, RFL and XFL, allows the neural network to learn patterns about which flights overfly certain areas.
  Reservation data has a time component. When constructing the grid cells in the training phase, the time at which the flight could theoretically reach the cell is calculated based on the expected entry time, expected entry point, the estimated aircraft speed and assuming a direct line. The estimated speed is derived from comparing expected entry times and exit times, and the distance between both points. Note that all parameters are taken from the data that would be known at the time the prediction is made, and may not be correct. But using actual data in the training phase would feed knowledge to the neural network about the actual trajectories, i.e. what it is supposed to predict, and hence would give false accuracy figures. In the final step, the theoretical time the aircraft could reach the cell is compared to the military reservation schedule

and the grid cell mapping. The upper FL for the cell is set accordingly.
The number of grid cells (more than 600, each with distinct values for upper FL) could lead to overfitting during the training phase. Overfitting is a common problem with machine learning. It means that the neural network starts memorising individual cases rather than generalising patterns. As a result, it achieves good accuracy on training data but poor accuracy on new data. To reduce the risk of overfitting, the intersection with the grid cells only considers a 120NM wide corridor between filed NCOP and filed XCOP. All other cells are set to 0. This approach reduces the risk of a unique fingerprint for each flight, but still provides all relevant cells to the neural network.

Initial versions of the neural network revealed a particular problem with the grid cells. Cells that do not intersect with military areas always contain the value 0 in the training set. When feeding real life data, some of the cells adjacent to military areas were assigned with FL values because of small differences in the geographic definition of military areas. The output of the neural network suffered from extreme errors under these conditions. The weights of the neural network are initialised with random values prior to training, as this is recommended practice to speed up training and avoid dead neurons. Because some input values are always zero, the weights linking them to neurons never contribute to the output of the neural network and, consequently, are not optimised during training.
The problem revealed a more general risk of overfitting: despite the large training data set, the possible values of some predictors may change in the future. The problem was addressed by initialising weights associated to grid cell data to zero and applying random noise during the training phase. The noise is injected for all input data unless the data is already uncertain by nature (e.g. entry times). Amplitude and standard deviation of the noise is configured per predictor. Random noise is superimposed for each training batch during iterative training.

The neural network

The neural network is implemented in TensorFlow[2], a framework for deep learning open sourced by Google. TensorFlow was selected because of its popularity, Python bindings, and the possibility to integrate it in production systems by means of a C++ and Java API.

A feed-forward neural network with 3 hidden layers containing 170 units each has been used. The last layer is connected to the readout layer with 8 units, corresponding to the 8 coordinate values to be predicted.

Hyperbolic Tangent , Rectified Linear Unit (RELU) and Exponential Linear Unit (ELU) (arXiv:1511.07289v5) activation functions have been evaluated. The ELU function appears to offer the fastest convergence time. The ELU function is applied on the outputs of the three hidden layers.

To reduce overfitting, a dropout is applied before the readout layer. It is used during training but turned off during testing. TensorFlow's dropout operator automatically handles scaling neuron outputs in addition to masking them.

[2] TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc. The authors are not affiliated with Google.

Cost function

The choice of a proper cost function is essential when training the neural network. The cost function expresses how 'good' the prediction is with respect to the real flown route, and allows the optimizer to adapt the neural network weights and biases accordingly.

The most correct cost function in view of the stated problem would be a calculation of the lateral distance between the position on the predicted route and the position on the real route at equivalent progression times. The maximum lateral distance in the time interval between AoR entry and exit would be the cost. This cost is referred to as $C_{lateral\_time}$.

Unfortunately, the use of $C_{lateral\_time}$ is difficult from a practical perspective:

- Both the predicted route and the flown route are sequences of angular line segments. It is difficult to express the lateral distance at equivalent times without resorting to conditional logic. Because the optimizer evaluates the partial derivatives of the cost function, a discontinuous cost function may not converge to a minimum during training.
- There is no time component in the predicted trajectory; one would need to assume an average speed along the length of the route.
- Early during training, there is no guarantee that the predicted route is similar to the flown route. The direction could be opposite, lengths could be completely different, or the segments could make U turns or intersect. The lateral deviation would be an incomplete cost metric in such cases.

A more pragmatic cost function consists of two components:

1. The sum of the distances between predicted and real entry points, and between predicted and real exit points. The distances give more weight to the lateral component due to the 0.5 scaling along the transformed X-axis.
2. The area of the polygon formed by the predicted route and the real route. The area is normalised by dividing by the length of the real route. The shoelace formula is used to calculate the area of the polygon. Although the formulate only gives correct results for simple (non-intersecting) polygons, it is useful as a cost function because the calculated area of intersecting polygons is always too high. During training, the optimizer will cause the network to avoid such routes due to their relative cost, in lieu for routes for which the shoelace formula yields correct results. Note that the area between the routes is not a proper metric if there would be lateral spikes in the predicted route. But with 4 predicted points and optimization of cost component (1), this is avoided.

$$C_{boundarypoints} = \sqrt{(xp_1 - xr_1)^2 + (yp_1 - yr_1)^2} + \sqrt{(xp_4 - xr_4)^2 + (yp_4 - yr_4)^2}$$

$$C_{area} = \frac{1}{2}|xp_1 \times yp_2 + xp_2 \times yp_3 + xp_3 \times yp_4 + xp_4 \times yr_4 + xr_4 \times yr_3 + xr_3 \times yr_2 + xr_2 \times yr_1 + xr_1 \times yp_1 - xp_2 \times yp_1 - xp_3 \times yp_2 - xp_4 \times yp_3 - xr_4 \times yr_4 - xr_3 \times yr_4 - xr_2 \times yr_3 - xr_1 \times yr_2 - xp_1 \times yr_1|$$

$$L = \sqrt{(xr_1 - xr_2)^2 + (yr_1 - yr_2)^2} + \sqrt{(xr_2 - xr_3)^2 + (yr_2 - yr_3)^2} + \sqrt{(xr_3 - xr_4)^2 + (yr_3 - yr_4)^2}$$

$$C_{pragmatic} = C_{boundarypoints} + \frac{C_{area}}{L}$$

Function $C_{pragmatic}$ is used to train, tune and test the neural network.

A variant of function $C_{lateral\_time}$ is used as final test of the integrated solution.

Optimizer and training

The choice of the optimizer has an impact on the convergence time. The AdamOptimizer algorithm (Kingma et al., 2014) with learning rate 0.0001 appears to be the best combination. The keep probability of the TensorFlow dropout operator is set to 0.8 during training.

2.600.000 iterations with batches of 1000 random fights are used. Longer training does not improve accuracy on test data but increases overfitting.

Validation of neural network output

When predicting routes for the flights of the cross-validation set with the trained neural network, the maximum lateral deviation versus the real flown trajectory is less than 6NM for 65% of the flights. This value is much better than the equivalent metric for the filed route (tagged 'EFD'):

Max distance from flown route	EFD trajectories	Neural Network trajectories
6 NM	10%	65%
10 NM	37%	84%
15 NM	60%	94%
30 NM	85%	99%

Figure 2 visualises the prediction for the flight that was used in figure 1. The predicted trajectory is in red. Figure 3 visualises the filed, flown and predicted routes for a sample with active military areas.

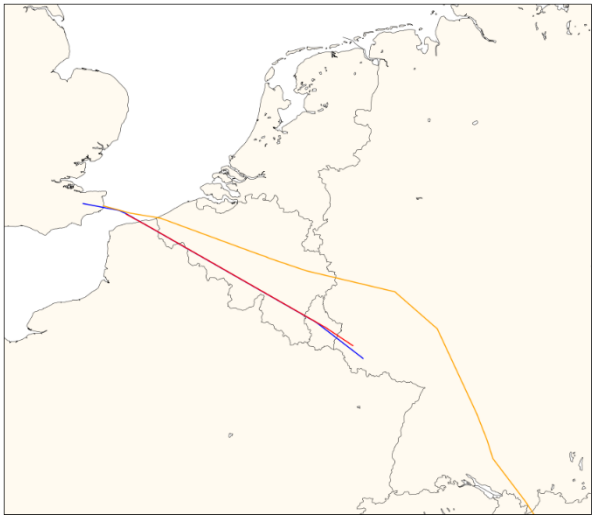

Figure 2: prediction (red) for flight of figure 1 (blue)

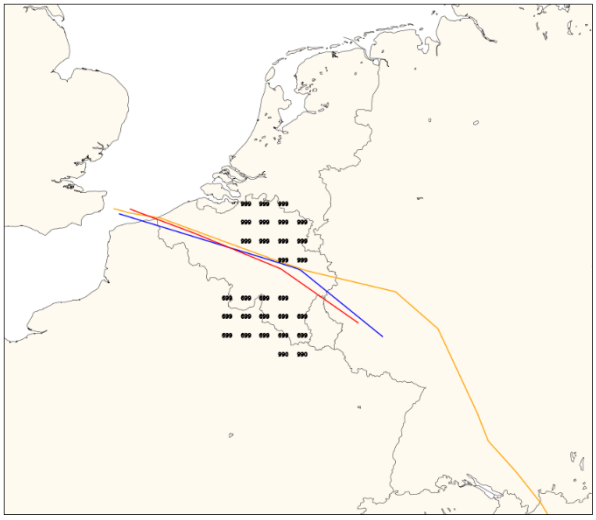

Figure 3: prediction for flight with active military areas

The deviation metric and visualisations allow assessing the performance of the neural network, and are very useful for hyper-parameter tuning, but do not perfectly reflect real life performance. The time element is not considered and correct knowledge about the predictors is assumed. In real life, the activation schedule of military areas may change up until the time of actual activation, the expected flight entry time could change due to departure noise, or the flight plan could be refiled with different parameters. Depending on look-ahead time, quality of the prediction will vary. At the end of the paper,

values of a variant of $C_{lateral\_time}$ are presented at different look-ahead times during real life operation. This metric reflects the full integration of the neural network in the operational system.

Integration of the neural network in the operational ATFCM system

After training, the model is serialized. TensorFlow saves the representation of the mathematical operations (the 'graph') and the learned weights and biases separately. A script is executed to merge the values of the weights and biases into the graph and remove all operators not needed for the inference phase. The resulting file is less than 1MB in size and is transferred to the operational system as adaptation data.

Because the new approach to trajectory prediction is fundamentally different from existing methods, and will bring many benefits in the pre-tactical phase when air traffic controller inputs are still unknown, the first operational implementation has been done in the ATM Flow & Capacity (ATFCM) system. An implementation in the Air Traffic Control system could follow the same architecture.

The Maastricht UAC ATFCM system consists of (a.o.):

- Flight Data Processing (FDP) system, responsible for trajectory prediction and sector sequence calculation in the planning phase, synchronized with the FDP system used in tactical operations. One of the items synchronized is military areas activation status and schedule
- Integrated Flow Management Position (iFMP) system, responsible for calculating traffic load and complexity metrics (occupancy & entry counts, weighted occupancy counts, clusters) and evaluating different airspace configurations for a given manpower planning.

The FDP system is fed with flight plan data for flights already under control and flights expected to enter the airspace in the next hours by means of EFD messages from the Network Manager.

Based on the flight plan data received from the FDP system, and the consolidation of airspace reservation data (TSA) from several sources (e.g. Military LARA system), the iFMP application continuously calculates the neural network predictors. If predictors change for a flight, iFMP invokes the neural network with a direct call to the TensorFlow Java API, which has been built into the iFMP application by means of Google's Bazel tool.

The iFMP application then calculates an appropriate branch-off point and rejoining point for the predicted route vis-à-vis the original route. Because these points are outside the AoR, the locations are less critical.

The resulting route is used to construct a What-If request for the FDP system. The What-If request triggers the FDP system to predict a 4D trajectory using its internal logic but constraining the route to the coordinates provided in the request. The What-If trajectory is maintained in parallel to the original trajectory and both are provided to iFMP system displaying them as traffic load to users (supervisors and flow managers).

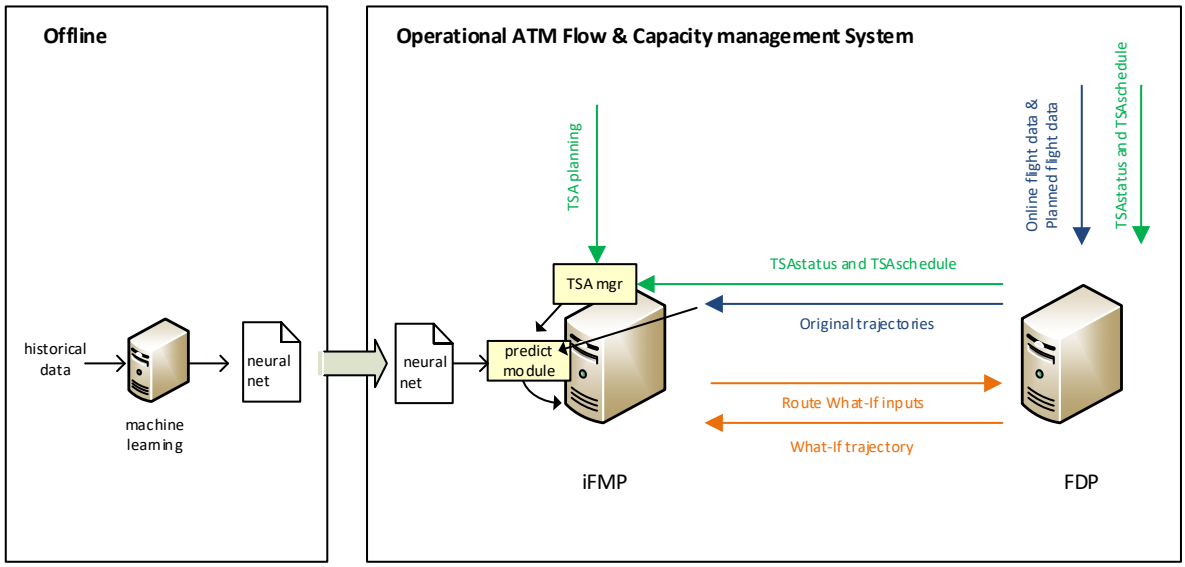

Figure 4: integration of the neural network in the production system

Flow and capacity management functions on iFMP use the What-If trajectory, for instance to calculate more realistic sector occupancy values. If problems would arise with the new prediction logic, iFMP can switch back to the original trajectories. What-If trajectories can be visualised and compared to the original trajectories.

The architecture offers following benefits:

- Novel techniques can be used in a system ranked safety related. It is not necessary to change the core of the FDP system and at any point in time, it is possible to switch back to unmodified trajectories.
- The approach allows merging machine learning with legacy prediction logic, and supports a roadmap to gradually replace other parts of the logic. E.g. predicting the climb or descent profile by machine learning, or predicting entry times.

Assessment of real life performance

Figure shows the accumulated lateral distances between the predicted trajectories and corresponding flown trajectories at different look-ahead times for the prediction (label: PREDICT). The same is done for the trajectories derived from filed flight plan data (label: RTTS).

Accumulated lateral distances at time $t_{look\text{-}ahead}$ are calculated as follows:

1. Consider the predicted trajectories available at time $t_{predict}$.
2. Get the aircraft position at time $t_{predict} + t_{look\text{-}ahead}$ on the flown trajectory (if in AoR)
3. For each aircraft position, calculate the lateral deviation with the predicted trajectory.
4. Calculate the mean of the lateral deviations.
5. Accumulate & normalise for all $t_{predict}$ in the evaluation period.

The figure is for 7 October 2017, and includes 543 flights from the UK to the east and southeast of Europe. Each box plot covers a rolled-up 1h look-ahead period. Lateral deviation is measured in meter.

The machine learned routes are substantially closer to the flown trajectory than the original ones.
The box plot denotes the 25%, 50% and 75% percentiles. The dotted tails are the outliers. Note that the

extreme outliers in look-ahead periods >4h are caused by refiled flight plans with different routes. They exist for both types of prediction.

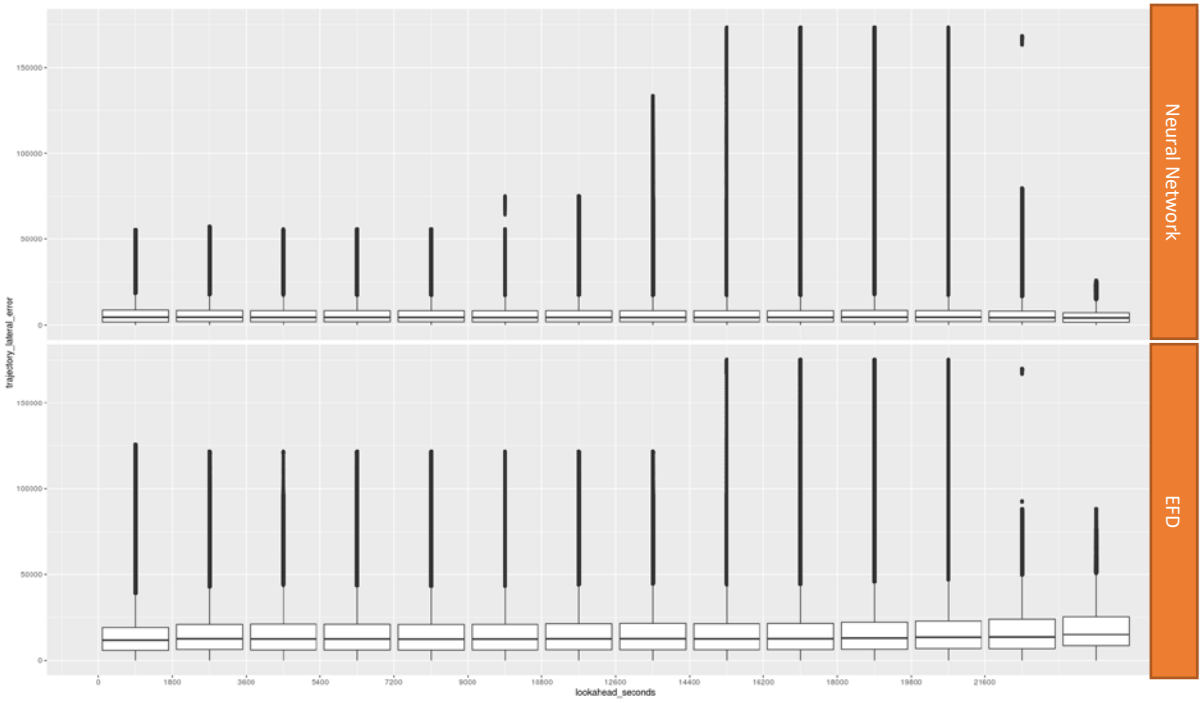

Figure 5: accumulated lateral deviation of predicted trajectory at various look-ahead times, UK departures 7 October 2017 to south and southeast European destinations

7 October was a Saturday with little military activity; conditions favourable for issuing direct routes. Figure shows the same statistics for Wednesday 6 September 2017, a busy day with 614 flights and several reservations of military areas.

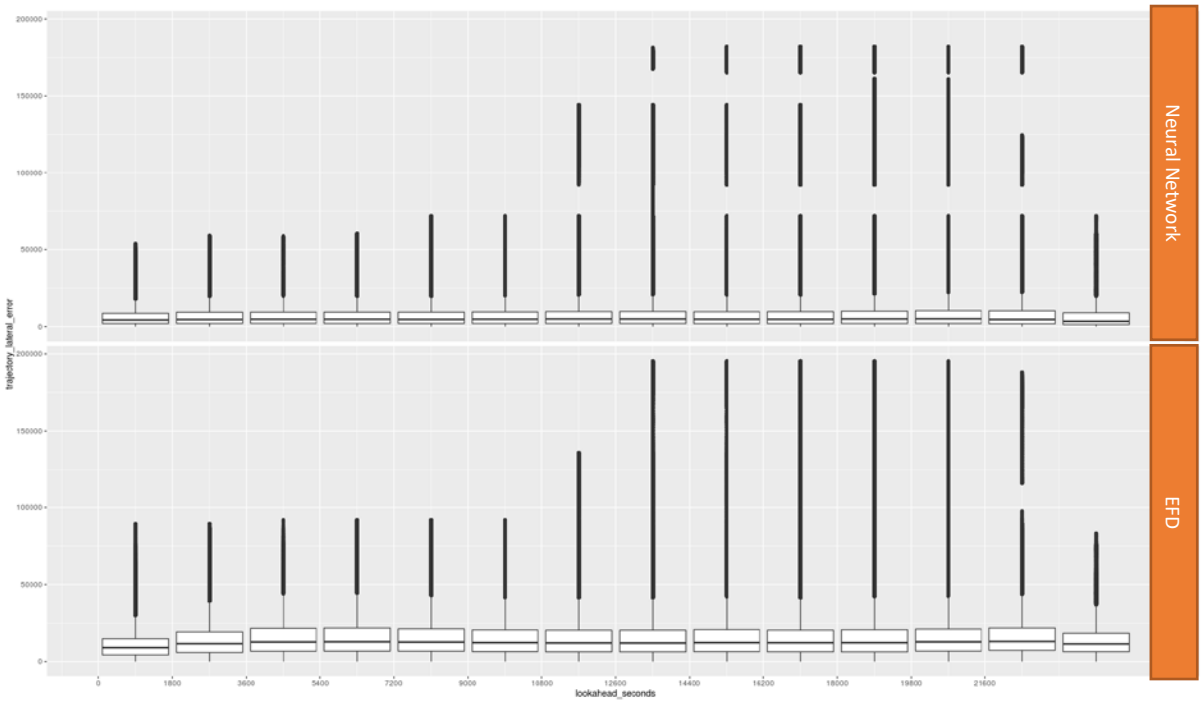

Figure 6: accumulated lateral deviation of predicted trajectory at various look-ahead times, UK departures 6 September 2017 to south and southeast European destinations

Figure depicts the consolidated statistics for all look-ahead periods up to 6 hours. The figure is for Friday 24 October 2017.

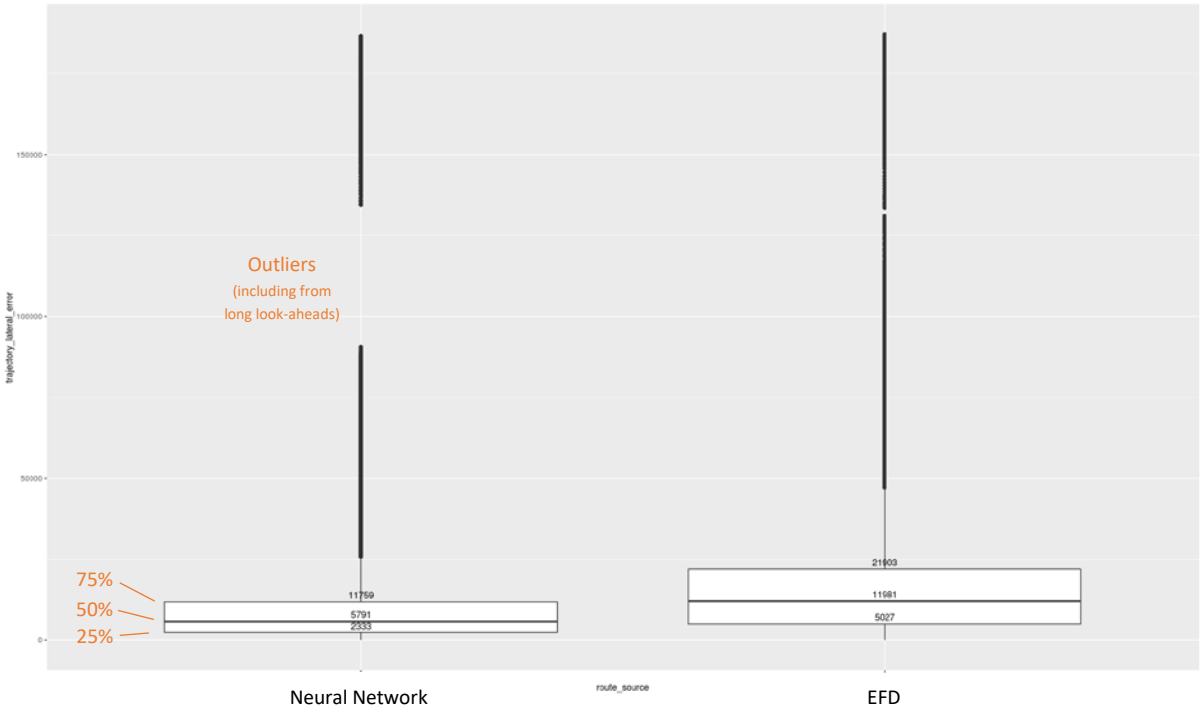

Figure 7: accumulated lateral deviation of predicted trajectory at all look-ahead times 0h-6h, UK departures 24 October 2017 to south and southeast European destinations

The figure shows that for the vast majority of flights the lateral error is reduced by half, without negative impact on outliers.

Conclusion and way ahead

The use of a deep neural network can produce flight route predictions that are substantially more accurate than methods in use today, which are based on data filed by the airline or pilot but do not take into account future air traffic controller clearances. EUROCONTROL Maastricht UAC has chosen to integrate the neural network in the existing 4D trajectory prediction logic, thereby combing the strengths of the legacy and the new logic and creating a path to gradually increase the application of machine learning technologies. The integrated solution is in use in the operational Air Traffic Flow and Capacity system for about 10% of all traffic. Results of the integrated solution show a clear improvement. As user confidence is built up, the scope will be increased to all traffic. A similar architecture may finds it way to the operational Air Traffic Control system.

Areas for future research are the prediction of the flight ascent/descent profile and the prediction of airspace entry times at longer look-ahead horizons. Existing trajectory prediction logic suffers from large errors in these areas.